COMX 35

使用手册

目　录

前　言

前　　言

我们自豪而满怀信心地向大家推荐本厂生产的 COMX－35 教学微型电子计算机。

COMX－35 教学微型电子计算机用途广泛，适合于学校教学，工商业，财会及家务等使用。它媲美一位博学多才的老师，可应付各种学术程度的需要。

COMX－35 教学微型电子计算机加上內附扩展设备，是一部价值相宜的多用途电脑，尤其适合于国内中学微型电子计算机教学使用。它经过精心设计，故使用极其方便。

COMX－35 教学微型电子计算机使用 BASIC 语言，有35K字节的随机存取记忆系统（RAM）（其中32K字节可供用戶使用）及16K字节的只读记忆系统（ROM）。配上本厂生产的显示器和储存器，成为一个完美的系统。它还可以加上其他设备如只读记忆，声音模拟器，打印机或记忆系统等以扩展其效能。

为了帮助用戶得心应手地使用 COMX－35 教学微型电子计算机，我们特编印本使用手册，它可以：

1. 使您不必读完整本手册就能被 COMX－35 的神奇性能所吸引，通过逐步解释，使您学会动手试验；

2. 说明计算机的一些基本概念，并详尽地为您介绍COMX－35 BASIC语言；

3. 提供许多程序设计的例子，进一步说明程序设计的方法。

本手册第一章逐步介绍如何起动计算机，如何与显示器，储存器，打印机等连接。

第二章介绍 COMX－35 的基本指令，并引导您学会使用计算机进行算术运算，作彩色图形，产生音响效果和做电子遊戏。

第三章将指导您作简单的程序设计，包括要计算机作判断和重复执行一段程序，并选择一些问题作为程序设计的练习，同时绘出答案，在您开始学习程序设计时，指导您养成进行程序设计的良好习惯。

第四，五，六章详细讲明算法，与程序的使用，介绍 COMX－35 的语言（BASIC），作图和音响效果等等，帮助您掌握计算机的基本概念和进行BASIC程序设计，帮助您成为称职的程序员。

有了 COMX－35 教学微型电子计算机再加上本使用手册，将使您提高智力，增进逻辑思维能力，激发您的想象力和创造力，对各类专业人员，学生都有极大裨益，尤其是学校教育中必不可少的打开奇妙计算王国的钥匙。

第一章

开机



第一章　开　机

1·1 打开COMX－35包装

COMX－35有下列组件：

(1)附加键盘和操作棒的COMX－35主机。

(2)一台西德乐满第显示器。

(3)一台COMX C－35卡式录音机。

(4)COMX－35P打印机。

(5)一根电源软线，一端是插头，另一端是交直流的换流器。换流器用于插进交流电源插座。请注意检查交流电源的电压是否与计算机要求的电压相匹配。

(6)一根显象电缆，一端是插头，另一端有一个联接器，用于插进电视机的天线输入插口。

(7)一对电缆（每端有二个插头）用于连接COMX－35和录音机。

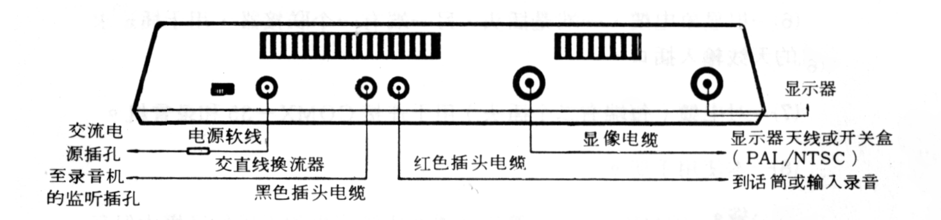
(8)一本使用手册。

请注意：COMX－35计算机必须在比较凉爽和干燥的环境中保存和操作。如果COMX－35准备长期搁置不用，请用一塑料袋装好，里面放一小袋硅胶。

1·2 接上显示器卡式录音机

显示器是作为显示设备使用，输入计算机的信息（按下键盘上的键）在显示屏上用一种颜色显示，从计算机输出的信息（例如计算的结果、图表等）在屏上用另一种颜色显示，卡式录音机是用来储存程序和数据的。关闭计算机时，暂时存在计算机存储器中的程序或数据就将消失，因此如果希望保存这些程序或数据，就应将其录进磁带。

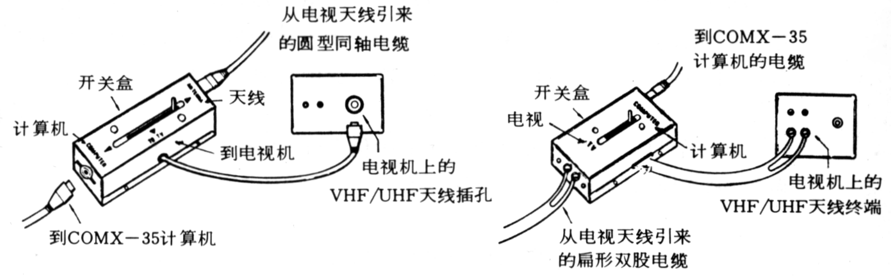
要将显示器与计算机连接，可将显象电缆末端的插头插进COMX-35背面标有TV PAL（或TV NTSC）的插口，并将联接器末端插进显示器的天线输入插口（COMX－35内部有频率调制器，故不必加入一个调制器）



要连接录音机，可使用所附的一对电缆，其中一根两头是红色插头，另一根是黑色插头。将一个红色插头插进录音机中标有 EAR 或 EARPHONE 或 MON 或 MONITOR 的插孔，另一个红色插头则插进 COMX－35 背面标有CASSETTE IN或EAR的插孔，再将一个黑色插头插进录音机中标有MIC或MICROPHONE的插孔；另一个黑色插头则插进 COMX－35 标有CASSETTE OUT或MIC的插孔。最后将录音机接上电源就可以使用了。

1·3 接上电源

使用圆形同轴天线电缆的安置法　　　　　　使用扁形双股天线电缆的安置法



首先，必须检查下列二件事：

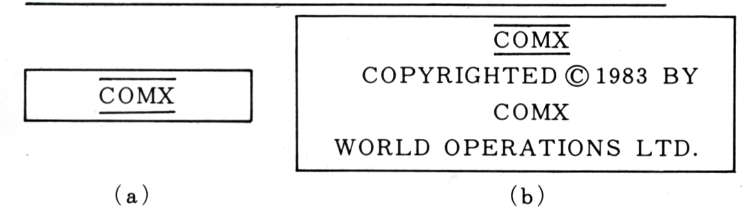
（a）COMX－35背面的电源开关是否确实处于OFF(关)的位置。

（b）检查电源换流器的额定电压是否与交流电源的输出电压相匹配。

然后，可将电源软线一端的插头插进 COMX － 35 背面标有POWER 的插孔，而另一端，即有换流器的一端插进交流电源插孔，再将电源开关拨到 ON (开)的位置。这时红色的POWER指示灯应发亮，同时可听到一些声音，它说明 COMX － 35 开始工作。

对PAL系统的电视机要调到36频道（UHF），对NTSC系统的电视机要调到了频道（VHF）（频道可能随不同的地方频率而变）然后将调到依次出现各种颜色的 COMX － 35 信息。至于调整 COMX － 35 计算机的问题请参考附录A·5

图1·3·1 COMX－35起动信息（随便按下一键就开始计算）



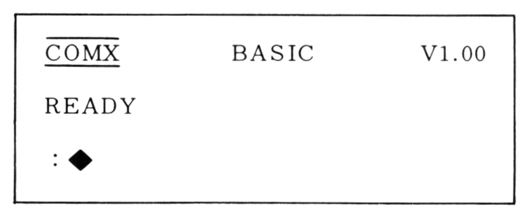
在调谐到COMX－35的电视输出频率时应该是缓慢地调。开始转动电视机上的调谐旋钮时，并没有出现电视画面，而是出现“雪花”要一直调至出现图1·3·1的信息。这时就可以调整电视机的对比度，亮度和色彩控制旋钮，可得到清晰的彩色显示，由于音响效果是来自COMX－35的內附扬声器而不是电视机的扬声器，故电视机的音量可调到最小。

起动信息(a)和(b)交替出现，其中信息(a)的颜色会变化，可以将它做为“测试模式”调整到最满意的颜色。

每台COMX－35都已经调整到有最好的颜色，如果沒有扰乱，一般只须对COMX－35底边上的一个 MIX 小孔作微小旋转就可调整好“测试模式”就是给做这种调整用的。

按下空棒外的任何键，计算机就开始进入计算状态，这时起动信息就为信息“READY”所代替。

图1·3·2 COMX－35“READY”信息



注意：显示屏的背景是黑色，计算机在屏上的输出是青色（浅蓝色），而菱形“光标”是粉红色。

光标指示下一个要显示的字符的位置。

冒号：是作为COMX 35 BASIC的“提示”符号（或简称为提示符），它提醒用戶计算机在等待你输入信息。

1·4 COMX－35 的键盘

在 COMX － 35 的键盘上共有包括空棒在內的55个键，其排列方式类似打字机的键盘。

SHIFT 键是考虑到很多键上面有两个字符而设的，如果单按某一个键，则显示屏上显示的是该键下面的符号。如果先按住 SHIFT 键再按此键则显示的是该键上面的符号。

与打字机键盘不同的是，COMX － 35 的键盘没有小写字母，另外，字母“O”和数码零“Ø”的区别就看此字母中间有无一斜线，要十分注意区别“O”与“Ø”和“L”与“1”。

要特别注意那些一般打字机键盘上所无的键的功能，开始时可能不完全了解，但这些键是十分重要。

CR 键（回车键）是使光标回到显示屏下一行最左边的位置（它同时送一个信息给计算机告诉它一条命令或指令已完。可参攷 2·1 节)CR 键也可简单地当作返回键。

RT（复原）键是使计算机恢复到初始状态，复原就是清除屏上的显示，并使计算机输出图 1·3·1 所示的信息，应注意，一旦同时按下复原键和空棒，记忆的內容就全部被清除，这意味着早先存放在存储器中的程序和数据全消失。

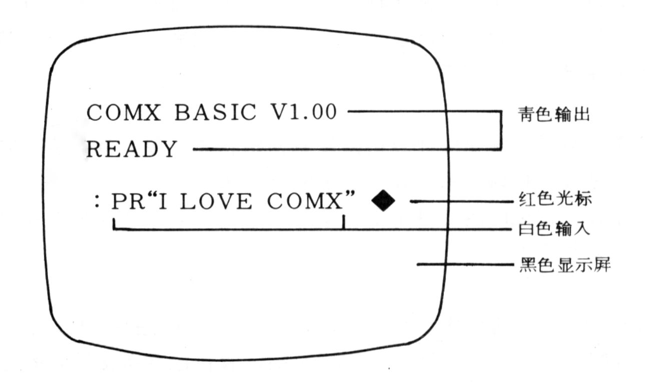
ESC（转义）键能中止程序的执行（从程序中“解脱”出来），当按下ESC键时，就显示信息“READY”说明计算机已准备接受新的 BASIC 指令。

单独按下CNTL（控制）键时，计算机什么事也不做，显示屏上也沒有任何字符，如果在按住CNTL键的同时按下另一个适当的键，计算机就相应地执行某些动作或打印出某些字符，例如，当按住CNTL时按下C键，计算机就不理正在打印的那一行，光标将移动到下一行最左边的位置。如果用戶在某一行出错而希望从下一行重新开始时，这个功能就很有用，CNTL键的功能将在2·1节和3·15节中进一步讨论。

1·5 COMX－35的显示屏

当电源初次接通时，计算机将用青色（浅蓝色）输出计算机产生的信息或计算结果的字符，而由键盘输入的字符则用白色在黑色显示屏（背景黑色）上显示，当然这是假定你使用的彩色电视机並且色彩、亮度和对比度都调整好，整个显示屏可显示24行，每行40个字符。

正如2·3节所要说明的，显示屏的颜色和字符可以用COMX－35 BASIC指令加以改变。



第二章

熟悉一些BASIC指令

第二章　熟悉一些BASIC指令

学完这一章，你就能够：

（a）将COMX－35当作一台高级计算器使用

（b）使用一些通用“BASIC”指令

（c）知道如果出错误，如何处理

（d）用“BASIC”指令绘彩色图表和谱曲

将 COMX － 35 当作计算器使用当然是没有充分发挥 COMX － 35 的无限潜力，但是，这是一种学习BASIC 语言的好方法，你可以使用这种操作方法做很复杂的计算，或帮助你的弟妹检查家庭算术作业，当熟悉了BASIC 指令之后，就可以使用一系列指令（即一个程序）去解决需要许多步才能解决的复杂问题，因此，这一章是为后面几章进行程序设计打好基础。



2·1 用COMX－35输出信息

首先启动计算机，得到READY信息。

然后在键盘上打　　PRINT“HI COMX－35”

接着再按C R键，在打引号(“)时不要忘记按下SHIFT键，按CR键是使光标移到显示屏的左边，同时也告诉计算机指令已完，这时屏上可看到：

你打的	PRINT“HI COMX－35”
COMX－35的应答	HI COMX－35

> 再说一遍，要输出信息，可使用指令 PRINT。后面接写由一对引号括起来的要输出的信息，再按CR键结束新指令。
>
> 可以不必打 PRINT，仅打其缩略形式 PR 也一样。
>
> 如果打了PR之后就按CR键，光标将移至下一行，这叫“换行”操作。

附注：如修正错误（或称编辑）

如果在按CR之前发现打错，可如下处理：

(1)按DEL（删除）键。将光标往后移这时被光标复盖过的字符将被删去，要改正，就用DEL键先删去错误的字符，然后重打。

⑵或者，按下CNTL键（控制键）再按C。这时就将光标移至下一行最左边位置，原来那一行（包含有错误）COMX－35认为它有毛病，不予理睬。

试打　　PRINT "TEST"

但在按CR之前，按下CNTL再按C。注意，这时沒有输出信息。因为对不完整的语句计算机根本不予理睬。

如果在按CR之后发现错误：

如果你将"PRINT"拼错会得到下面的错误信息：

```
ERR CODE 30
READY
:◆
```

代号30的错误信息的意义是。在PRINT语句中最后一个字符是不可接受的，冒号：说明原来並未受影响，你可以重打指令。比如你打PRNNT "HI" 其中有一个"N"是"I"之误，计算机仍然认出头两字母"PR"它与"PRINT有同样意义、然后计算机将NNT"理解为一个变量，而将"HI"作为文字。但在它们之间沒有逗号或分号之类的分隔符，由于缺少分隔符故造成错误。

如果沒有前面或后面的引号，计算机就输出信息如下：

```
ERR CODE 22
READY
:◆
```

代号22的错误信息的意义是"缺少引号"

其他错误信息及其意义可见附录。

如果你打了指令之后没有得到任何应答，那也许是忘记在指令的结尾按CR键。

例：

试试下列指令看是否得到相应的输出。

```
打→  PRINT"MONEY CAN'T BUY ME LOVE—BEATLES"
     MONEY CAN'T BUY ME LOVE—BEATLES

打→  PRINT"TO ERR IS HUMAN, TO REALLY MESS THINGS
     UP YOU NEED A COMPUTER"
     TO ERR IS HUMAN, TO REALLY MESS
     THINGS UP YOU NEED A COMPUTER

打→  PRINT"A CHILD MISEDUCATED IS A CHILD
     LOST-JOHN F. KENNEDY"
     A CHILD MISEDUCATED IS A CHILD LOST-
     JOHN F. KENNEDY.
```

用PR代替PRINT，再重复上例，並注意计算的应答是否一致。

2·2 将COMX－35作为计算器使用

COMX－35可作为高级计算器使用

在COMX－35上试打

```
PRINT   5 + 6
```

並按CR键，COMX－35将应答明：

```
11
```

COMX－35能进行5种算术运算：

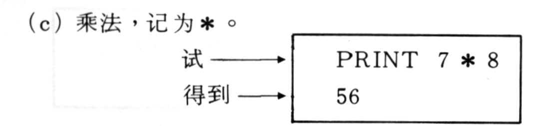
(a) 加法，记为十。

(b) 减法，记为一。　试：

```
PRINT  34－16
```

得到18. 在指令的末尾不要忘记按CR键。

(c) 乘法，记为＊。



试 ——→ PRINT 7 ＊ 8

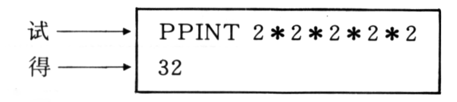
得到 ——→ 56

(d) 除法，记为／。

试 ——→ PRINT 56／7

得到 ——→ 8

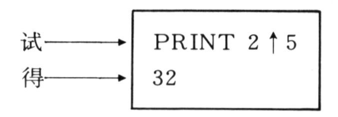
(e) 幂。有时需要将一个数自乘给定的次数。



试 ——→ PPINT 2＊2＊2＊2＊2

得 ——→ 32

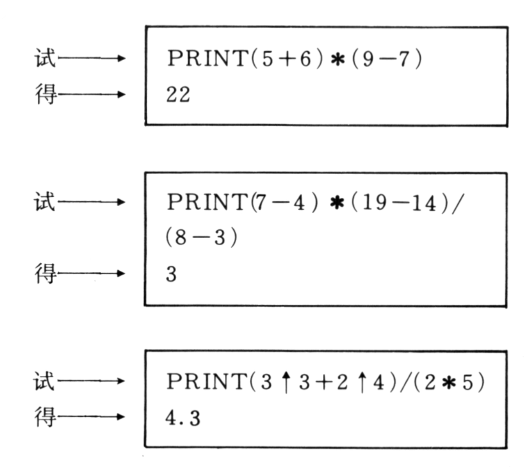
不过，上述的写法可换成下面的写法：



要打出向上的箭号↑须在按下 SHIFT 键时按 △ 键。在设计“ BAS-IC ”一类计算机语言时，对某些函数都规定特定的符号，故使用时不能用错键。这些特定符号都可在计算机键盘上找到，例如（ 3 × 5 ）中的×变成＊。因此做乘法就不能用“×”。除法的“÷”变成“／”，在普通键盘上没有“÷”、“＋”号和“－”号保留不变。由于取幂运算要在字符的右上角放一表达式，如“ 10^2 ”，而大部分计算机都不能表示这种“上标”字符。因此需要有一个符号，结果选择“ ↑ ”，在它后面的字符即是指数（例如10^2写成10↑2）。

上面的例子都包含两个数和一种运算。

COMX－35还可解答更复杂的问题，其中含有多个数及若干种不同运算。



计算机进行不同运算的顺序类似算术中的普通法则。即先做*和／，再做＋和－。在5·3节就给出完整的顺序表，这里使用括号“（　　）”与通常一样，是告诉计算机其中的运算应该先做。

2·3 改变颜色

COMX－35屏幕显示是多颜色的。用戶的键盘输入用白色显示，计算机输出及错误信息用青色（浅蓝色）显示。菱形光标是粉红色。这样做的目的是使得更醒目，便于用戶与计算机之间的“对答”，提高用戶编程序的速度和减少错误。这对于初次使用者显得更重要。上述的颜色组合可说是“缺乏”选择的，即是说一般情况下COMX－35都选用这些颜色，除非有特殊指令要求使用其他颜色，通过使用下面要说明的COLOR，SCREEN，和CTONE等指令就可以选择其他颜色。

2·3·1 试验COLOR指令

COLOR指令的形式是：

COLOR(X)

其中X是数表达式，由它规定用戶输入和计算机输出字符所要使用的颜色组合。表2·3·1列出不同的X值所对应的颜色组合。

现在打指令

```
COLOR(10)
```

再按CR键，注意到用戶输入的字符已变成粉红色而计算机输出变成蓝色。再试试其他一些颜色指令，将“10”换成1到12中任一数，观察其颜色的变化並与表2·3·1做比較。注意COLOR(12)就相应于原来的字符颜色。

表2·3·1 对应COLOR(X)的颜色选择

X	计算机应答		键盘输入	
1	黑色	(0)	绿色	(59)
2	红色	(30)	黄色	(89)
3	蓝色	(11)	青色	(70)
4	洋红	(41)	白色	(100)
5	黑色	(0)	蓝色	(11)
6	红色	(30)	洋红	(41)
7	绿色	(59)	青色	(70)
8	黄色	(89)	白色	(100)
9	黑色	(0)	红色	(30)
10	蓝色	(11)	洋红	(41)
11	绿色	(59)	黄色	(89)
12	青色	(70)	白色	(100)

注：括号中的数字是光亮度(%)

2·3·2 试验SCREEN指令

SCREEN指令的形式是

SCREEN(X)

其中X是数表达式（可取1到8中任一数），由它规定显示屏的背景颜色。表2·3·2列出不同的X值所对应的显示屏的颜色。

现在试试指令

SCREEN(2)

再按CR键。注意到屏上颜色由黑变为绿，用1到8中任一数代替2，并将屏的颜色与表2·3·2比较。注意，SCREEN(1)就相应于原来的屏色，还要注意，如果输入字符或输出字符的颜色与显示屏的颜色相同，则字符就看不出來。

在表2·3·1中还给出COMX－35不同颜色的光亮度，一般说来，用明亮的字符对灰暗的背景就能得到满意的显示（例 COLOR (12) 和 SCREEN(1)，或COLOR(12)和SCREEN(3))；也可以使用灰暗的字符对比明亮的背景。(例如 COLOR(5)和SCREEN(2))；COLOR(12)和SCREEN(7)也令人感到比较舒服。

表2·3·2 对应SCREEN(X)的屏色

X	颜色	
1	黑色	(0)
2	绿色	(59)
3	蓝色	(11)
4	青色	(70)
5	红色	(30)
6	黄色	(89)
7	洋红色	(41)
8	白色	(100)

注：括号中的数字是光亮度(%)

2·3·3 CTONE指令

其形式是

CTONE(X)

其中的数表达式X只取零与非零，如果X非零，其色调就变成这样：屏色不变，而显示出来的字符的颜色类似于屏色，但亮度不同，如果X是零，则沒有上述的色调效应。

2·4 产生音的效果

在COMX－35中，可通过内部扬声器产生高质量的音响效果。“音乐指令”可产生8个音阶频率范围和16个响度级的七个音符。“声音指令”可产生8个音阶频率范围和16个响度级的高斯白噪声。另外，有“音调指令”。它类似于音乐指令（即有8个音阶和16个响度级），只是它对每个音阶，而不是七个音符，其频率可有128个音级的变化。适当地配合使用这三个指令就可以谱成一首乐曲，同时使逼真的音响效果与图样和颜色的变化相陪衬。

2·4·1 试验MUSIC指令

MUSIC指令的形式是

MUSIC(X,Y,Z)

其中X,Y和Z是数表达式，X规定音符，可取1,2,…到7。即是

1＝DO
2＝RAY
3＝MI
4＝FA
5＝SO
6＝LA
7＝TI

Y决定音阶，可以1,2…变到8，8是最高频率的音阶。

Z决定振幅，可从0，1变到15，15是最响。

现在试：

```
MUSIC(1,1,1)
```

再按ＣＲ键，可得最低音阶的微弱的"DO"

再试： MUSIC(1,1,2)

注意到音量提交了。

再试： MUSIC(2,1,2)

注意已变成"RAY"

再试： MUSIC(2,2,2)

注意到音阶变高了。

应注意一发生音乐指令，该音符就一直持续到发出另一条音乐指令为止，要取消此音符，可打

MUSIC(0,0,0)

当然，在一条指令的末尾总是不要忘记按ＣＲ键

2·4·2 试验NOISE指令

NOISE指令的形式是

NOISE(Y,Z)

Y规定高斯白噪声的频率范围，可从1,2,……到8

Z规定振幅，可从0,1,2,……变到15。

试试 NOISE(1,1)

再按CR键，並注意声音效果

再试：

```
NOISE(1,2)
```

注意其音量（或振幅）提高。

再试：

```
NOISE(2,2)
```

注意其音调（或频率）提高。

要停止发声音，可打

```
NOISE(0,0)
```

注意，在每一指令的末尾总是要记住按CR键。

2·4·3 试验 TONE 指令

TONE 指令发出一个连续音调，其形式是

TONE(X,Y,Z)

其中X规定频率，可从1,2,……变到 128，

Y 规定音阶，可从1,2,……变到 8

Z 规定振幅，可从0,1,……变到15。

表 2·4·3列出实际输出频率和TONE指令中参数X之间的关系，该实际是对第 4 音阶（即Y＝4）对更高音阶其频率应加倍，而每低一音阶其频率减半。

表2·4·3

参数 X	频率 (Hz)	音符	参数 X	频率 (Hz)	音符
46	482	B	49	452	B^b/A#
52	426	A	54	410	A^b/G#
58	382	G	61	363	G^b/F#
65	341	F	69	321	E
73	303	E^b/D#	77	287	D
82	270	D^b/C#	86	257	C

2·4·4 VOLUME 指令

其形式是　　VOLUME(X)

其中整数X可从1（最小声）变到4（最大声）这个"总音量控制"指令对其后面所有发出的MUSIC，NOISE和TONE指令都起作用，在其前面的所有音响指令就不受其影响。

2·5 用录音机录存程序和数据

2·5·1 用PSAVE将程序从计算机转录入录音机

首先，按照1·2章所说的步骤连接好录音机。

假定在计算机存贮器中有一个希望保存起来供今后使用的程序，应先打LIST仔细检查在存贮器里的程序中有无不需要的程序片段、再调整录音机的音量控制到适当位置，对COMX－35来说，通常置音量控制于中间位置，现在，依次如下操作：

(1) 往录音机装上空白磁带並反绕此带（如果录音机上有磁带位置计数器，应让它置零。当然，如果使用已用过的磁带进行录音时原来已录存的信息会被抹去。）

(2) 同时按下"PLAY"和"RECORD"开始在磁带上录音。

(3) 打PSAVE（录存程序）再按CR键，开始将程序录进磁带。

(4) 当(3)完成时，就重新显示信息"READY"，后面跟"：◆"请不要忘记记下被录程序终了时的磁带位置，以备今后查找。

进行第(3)步的详细过程是这样的，当打了PSAVE並接着按CR键之后，光标就消失，同时可听到标志程序开始（称为程序头）的一阵尖叫声。

我们应该先让录音机转动以保证不会录到空白的带头上，一般磁带的带头是不能录下任何信号的，因此假如从带头开始录的话就可能会丧失一部分程序或数据。

然后是一声噪音，表示在传送程序的第一页（256位组）接着是一阵短促的尖叫声，这是页头，标志第二页开始，以后又是噪音表明在传送程序的第二页。往后，尖叫声和噪音交替出现一直到整个程序传送完毕，传送结束的标志是尖叫声之后投低音，熟悉上述各种声音的出现顺序之后，用戶就随时能知道已传送到什么地方。

COMX－35是这样设计的，电计算机发出声音信号经由联接两个MIC插孔的电缆到录音机、信号到达录音机之后再经由连接两个EAR插孔的电缆返回计算机，如果有错误或沒有连接好，或在录音机未运转，都导致信号迴道不通，也就听不到声音，这就提醒用戶应检查连接的情况。

* 应该注意，不是所有的录音机在录音时都能产生上述的声音，如果有这种情况也不必担心，还是可以用它与COMX－35连接使用，只是不能听到声音罢了。

2·5·2 用 DSAVE 录存数据

要录存数据，步骤(1)与(2)与上面一样，仅是将(3)中的PSAVE改为DSAVE（数据录存）即可。这时计算机中用于存数数据（数字符串式数组）的那一部分存储器的內容就被存到磁带中。

与上一节所说的一样，也会发出一系列声音表明录存过程的状态。

2·5·3 用 PLOAD 将程序从录音机回转给计算机

为了将录存在磁带上的计算机程序回输式装入计算机存储器，首先应将录音机如1·2 节所说的联接好，然后按以下步骤操作：

(1) 往录音机装上磁带，並找到程序开始的地方。

(2) 弄清楚存储器中现存的程序或数据是否已不再需要，或者已经录存起来，因为进行第(3)步时就会清除存储器的全部內容，原来存在存储器中的程序或数据会全被洗掉。

(3) 假定计算计已接通电源，同时显示出“READY”信息和“：◆”。现在打PLOAD，並按下PLAY按钮（在录音机上），当听到一阵尖叫声时，就按ＣＲ键开始回转程序，这时光标消失，並伴有2·5·1即所说明一系列声音。

(4) 当回转程序完成时，信息“READY”和“：◆”重新显现。

2·5·4 用 DLOAD 回转数据

DLOAD指令，（数据装入）是将先前（由指令DSAVE）存入磁带上的数据回转给计算机。其过程类似2·5·3所述，只是将PLOAD改为DLOAD。这时自动地将数据放在现存的程序存储空间的末尾並冲掉原有的数据。

2·5·5 关于 DLOAD 的另外几个要点

关于 DLOAD 指令的用法，有三点必须考虑的：

(1) 如果用戶的存储空间同时包含有程序和数据，对程序的任何编辑都会把数据抹去，因此在做编辑之前，应先用DSAVE录存数据（如果希望保存的话），然后才对程序进行编辑，最后用DLOAD回转数据。

(2) 注意，做DLOAD的作用是自动定义一节大小不超出所要存的数据的数组。

(3) 字符串都存放在数组空间的末尾，如果在字符串已生成还要再定义数组，则由于数组空间所加会将这些字符串抹去，因此在字符串生成之前应将所有数组全定义好。

2·5·6 关于在磁带上录存和回输程序的一些附言

为了防止丧失有价值的程序和数据，下面几点是应该注意的：

(1) 可反复使用 PSAVE 和 DSAVE 来录存多项资料，对于需进行几小时工作的程序，可同样录存在另一磁带上。

(2) 留心听声音序列，它是成功进行传输的主要标志。

(3) 第(1)步之后，将已录存的程序回输给计算机并 LIST（列程序清单），如果回输不能顺利完成，原来程序仍可能完整无损，故还可以重新想法录存程序。

(4) 拿磁带应小心防止抓伤，还应让磁带避开强磁场，保存磁带应远离电视机，其他注意事项可见附录A3。

第 三 章

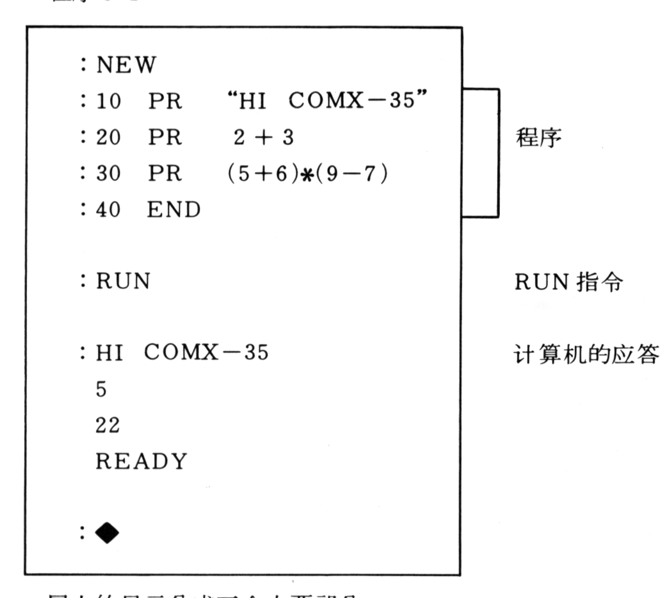
编写简单程序

第三章　编写简单程序

3·1 第一个 COMX－35 BASIC 程序

下面一些例子来介绍程序设计，先打进这些程序例子，然后观察计算机的应答，第一个程序是用来告诉你一个程序的主要结构及某些BASIC指令的用法，打下列程序：

程序 3·1



属上的显示分成三个主要部分。

(i) 程序本身

(ii) RUN 指令告诉计算机去执行程序的指令。

(iii)计算机将执行程序指令的结果输出。

下面进一步看工程本身。

NEW是指令(或语句)，它告诉计算机清除以前输入的全部程序或数据。

程序中每一行指令以一个行号开始，如10，20等等，计算机将依行号的顺序，即从最小到最大依次执行各指令。

当你看到指令10、20和30时，就会注意到这些第二章已遇到过，不过在这里，这些指令是作为一群来执行(在打RUN之后)，而不是一次执行一条指令，这是将COMX－35作为运行程序的计算机和将COMX－35当作计算器使用，这两者之间的主要区别。

指令END表示程序结束，当计算机看到END时，就停止程序的执行。

RUN是指令，它告诉计算机开始执行程序，计算机找出最小行号然后依行号顺序开始逐行执行下去。

附注：关于行号的一些问题

建议你用行号10来开始你的程序，按下去是20、30、40、……等等，即留下9个间隙可放其他指令，这将使你能不改变原来行号而插入额外增加的行，比如你加入行15，这条指令将在行20之前执行。

当你用行号来开始指令时，实际上正是告诉计算机在打RUN之前不要执行它，那是在编程序状态下使用计算机，如果不使用任何行号，就象第二章一样，则只要一按CR键，马上就开如执行，因此，是处于作为计算器的状态。

行号也可看作是指令的名字(或称标号)。在GOTO语句中和程序编辑时也使用行号，见3·2节和3·3节。

COMX－35提要：什么叫程序？

程序是一组（或一序列）指令，其中每一条指令都由一个行号开头，一般情况下，计算机将按行号从小到大的顺序执行这些指令，程序开头的NEW指令是用来清除原先的程序，而指令END是令计算机停止执行，指令RUN告诉计算机从最小行号的那条指令开始执行，指令運常也称为语句，因此我们也说语句END，语句RUN等。

3·2 使计算机跳转（使用GOTO语句）

正象3·1节所解释的，一般情况下计算机是依次执行各指令，然而，也可使用GOTO语句让计算机不按序执行指令，请在3·1节的程序中插入语句"15 GOTO30"得到：

程序 3·2

```
NEW
10    PR       "HI  COMX 35"
15    GOTO     30
20    DR       2 + 3
30    PR       (5+6)*(9-7)   P
40    END
```

附加GOTO语句

RUN

```
HI  COMX-35
22
```

注意：语句20幷随之其输出"5"被跳过。

COMX－35提要：GOTO语句

语句"GOTO n"告诉计算机下一步不再执行此语句的后续语句，而是去执行行号 n 的语句，这种语句叫做"无条件转移"因为它不管什么条件，总是让计算机跳转或转移到另一语句，如果没有写上行号，就会产生错误信息。"n"可以是表达式。因此，象"GOTO A-B"和"GOTO 50*(A＋B)"都是允许的。

3·3 显示和编辑程序

下面假定已将3·2 节中的程序打进计算机。如果还未有的话，请在做下面处理之前先打进去。

为了使计算机显示程序，请试打：

```
LIST
```

并接着按ＣＲ键，这时计算机将显示出刚打进去的程序，在本例中是程序3·2。

```
10    PRINT    "HI  COMX－35"
15    GOTO     30
20    PR       2＋3
30    PR       (5＋6)*(9－7)
40    END
```

若要更替某一行，可先打要更替的那一行的行号，然后打进新的指令即可，例如要更替行15的内容，试打：

```
15    GOTO     40
LIST
```

计算机显示出：

```
10    PRINT    "HI  COMX-35"
15    GOTO     40
20    PRINT    2+3
30    PRINT    (5+6)*(9-7)
40    END
```

若要删去一行，可先打此行号，然后按ＣＲ键即可，例如要删去行15，试打：

```
15
LIST
```

行号之后要按ＣＲ

计算机将显示出：

```
10    PR     "HI  COMX-35"
20    PR     2+3
30    PR     (5+6)*(9-7)
40    END
```

若要在行20和行30之间插入一行，可选一行号，比如说25，然后打:

```
25    GOTO    40
LIST
```

计算机将显示出：

```
10    PR      "HI  COMX-35"
20    PR      2+3
25    GOTO    40
30    PR      (5+6)*(9-7)
40    END
```

新的行已插入

COMX－35附注：关于LIST的几种形式

LIST
LIST n
LIST n, m

语句 LIST（后面不跟任何符号）是令计算机显示或列出全部程序，“LIST n”仅要求列出行号n的那一行，“LIST n, m”要求从行n开始列到行m为止，“n”和“m”都可以是算术表达式。

如果表达式的值不是程序中某一行的行号，那么计算机将列出此值后面最接近的行。

COMX－35提要：编辑程序

所谓编辑程序就是改正或修改程序，可通过使用行号来更替，删去或插入某一行，并总是使用LIST来列出该程序或被编辑过的那一部分，以便确信是否如希望的那样被修改好（也可参见3·11节）

3·4 编写一个能从键盘接受数据的程序（使用INPUT语句）

从3·3节中最后一个程序出发，试打：

```
5          INPUT  A,B
10
20         PRINT  A+B
25
30
40         END
```

插进新的INPUT语句删去行10

删去行25和30

```
LIST
```

经过这样编辑之后，计算机将显示出：

程序 3·4

```
5          INPUT  A, B
20         PRINT  A+B
40         END
```

行5是输入语句，A和B是"还未指定的"或"未知的"量，称为变量，当计算机遇到INPUT语句时，就停下来并在显示屏上输出一个问号"？"，然后等待回答用戶，用戶应打进与变量个数相同的若干个数，这些数就做为这些"未知量"或"变量"的值。

例如 A＝2， B＝3 试打：

```
RUN
?          2,3
5
```

用戶在回答"？"时输入数据作为A,B的值。

计算机输出，等于A＋B

对于A =1012， B =4517　试打

```
RUN
 ?          1012， 4517
5529
```

A 和B 的另一对数据A + B

现在已经有了一个当给出二个数量的值就能得出它们的和的程序。

请试：A 和B 另外的值。

下面，用更复杂的表达式来替换行20

```
20        PRINT    (A*A+B*B)↑0.5
LIST
```

程序变成

```
 5        INPUT    A,B
20        PRINT    (A*A+B*B)↑0.5
40        END
```

执行此程序，给出

```
RUN
 ?        3,4
 5
```

你可能已认出上述程序就是大家在几何中所熟知的勾股定理的应用，如果你想更详细了解此定理，请读下面的脚注，然而，重要之处在于说明这三行程序的巨大作用。

可以将行 5 修改成能为多个变量接受数据，行20可用这些变量的更复杂的代数表达式来替换，执行此程序时，在回答“？”时就给这些变量赋值，计算机就输出此表达式的结果，你可以用另外一组数值重新计算而无需再打进此表达式，你输入数据，计算机就执行程序给出答案，这就是计算机程序的功能的眞髓，可用下图说明之：

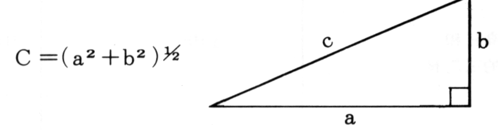
输入数据 → 计算机程序 → 输出答案

在不带行号而使用计算机时（如第二章所说的，即所谓计算器方式），对每一组数据都应提供计算的步骤。

COMX—35 附注：勾股定理

若直角三角形的斜边长为C，另二边的长为 a 和 b，则按照勾股定理有：

$$C=(a^2+b^2)^{1/2}$$



就是说，给出 a 和 b，就可为上式计算C，为了用 BASIC 做到这一点，应该将左边代数表达式改写为 BASIC 表达式，即（A＊A＋B＊B）↑0.5，下一节将再给出这样一些如何将代数表达式改写成 BASIC 表达式的例子。

3·5 代数表达式和BASIC表达式

你可能对某些代数公式很熟悉，例如计算圆的周长和面积的公式，然而，由于计算机只能认得BASIC表达式，故在将这些代数公式输入计算机之前就应该将它改写成等价的BASIC表达式，举例来说，“ a b ”，“ a ÷ b ”，“a^b”应该相应写成“ a * b ”，“ a ／b ”，“ a ↑ b ”。下面的表给出另外一些例子：

表3·5 将代数表达式转换成BASIC表达式

说　明	代数表达式	BASIC表达式
1. 半径为R的圆面积	πR^2	P1 * R ↑ 2
2. 半径为R的圆周长	$2\pi R$	2 * P1 * R
3. 给定高H和底B的三角形面积	$\frac{1}{2}HB$	H * B/2
4. 给定三边A，B，C（和$S=\frac{1}{2}(A+B+C)$）的三角形面积	$\sqrt{S(S-A)(S-B)(S-C)}$	S=(A+B+C)/2 (S * (S−A) * (S−B) * (S−C))↑0.5
5. 给定直角三角形二边A, B求斜边C	$C=(A^2+B^2)^{1/2}$	C=(A * A+B * B)↑0.5 或 C=(A↑2+B↑2)↑0.5
6. 给定二次方程$AX^2+BX+C=0$求解X	$X=\frac{-B\pm\sqrt{B^2-4AC}}{2A}$	X=(−B+(B * B−4 * A * C)↑0.5/(2 * A)和 X=(−B−(B * B−4 * A * C)↑0.5/(2 * A)
7. 年复利率为R％，求为P，n年后的“本利和”	$P(1+R)^n$	P * (1+R)↑n

依照3·4节的3行程序的模式的表3·5最右列，试写出全部七个问题或其中若干个的程序。

3·6 让 COMX－35 做判断

依次打进下面计算三角形面积的程序，RUN，並输入三组数据如下：

程序 3·6

```
NEW
10      INPUT A,B,C
20      IF A<0 THEN GOTO 50
25      S=(A+B+C)/2
30      PRINT(S*(S-A)*(S-B)*(S-C))↑0.5
40      GOTO 10
50      END
RUN
?       7,8,9
26. 8328
?       12,14,17
83. 0267
?       -1,2,3
READY
:◆
```

当计算机遇到行40的语句“GOTO 10”时，它总是转回到语句10並询问另一组输入数据，这样可以无限制地进行下去（假定给定的数据是正确的，即三角形任二边之和大于第三边），用这小小的程序就可以计算世界上所有三角形的面积！问题倒是如何叫 COMX－35 停下来！那是行20的目的，这个新的语句要求计算机检验A的值，如果A是小于0，那末转移到行50从而中止程序的执行，这个“IF…THEN GOTO”语句称为条件转移，就是说，如果某个条件为眞时就转移。

也可说 COMX－35正在根据某些条件决定下一步做什么。

COMX－35附注：IF语句

符号

＞　大于
＜　小于

称为关系算符，其他的关系算符是

＝　等于
＜　＞不等于
＞＝　大于或等于
＜＝　小于或等于

在很多BASIC版本中，"IF…THEN GOTO"语句是"IF"语句的唯一形式，但在COMX－35 BASIC中，"IF…THEN"后面可接一组语句（用冒号分开），这是很重要的改进，更加实用，使程序更容易理解和"构造"。

行25是赋值语句，将右边表达式的值给予或赋给左边的变量。

应用IF语句的这第二种形式，上述程序可改写如下，但仍给出同样结果。

```
NEW
10    INPUT A,B,C

20    IF A>0  THEN  S=(A+B+C)/2:
      PRINT(S*(S-A)*(S-B)*(S-C))↑0.5:GOTO 10

30    END
```

行20，包含用冒号分开的多个语句，仅当A＞0时，才执行这些语句，如果A＜0，就跳过行20，到行30执行停止。

3·7 形成搞程序设计的好习惯

让我们来改进程序3·6，打下列各行：

程序 3·7

```
NEW
1     REM THIS IS A PROGRAM TO COMPUTE THE
      AREA OF A TRIANGLE
5     PRINT"THIS PROGRAM COMPUTES THE AREA
      OF A TRIANGLE"
10    INPUT"PLEASE INPUT THE LENGTHS OF
      THE SIDES A,B,C" A,B,C
20    IF A<0 THEN GOTO 120
30    S=(A+B+C)/2
40    IF S<A THEN GOTO 100
50    IF S<B THEN GOTO 100
60    IF S<C THEN GOTO 100
70    T =(S*(S-A)*(S-B)*(S-C))↑0.5
80    PRINT"THE AREA OF THE TRIANGLE=";T
90    GOTO 10
100   PRINT"INCORRECT DATA:TWO SIDES OF
      THE TRIANGLE NOT LONGER THAN THE
      THIRD"
115   GOTO 10
120   PRINT"END OF SESSION"END
RUN
```

```
THIS PROGRAM COMPUTES THE AREA OF A
TRIANGLE
PLEASE INPUT THE LENGTHS OF THE SIDES
A,B,C ? 12,13,14
THE AREA OF THE TRIANGLE=72.3079
```

```
PLEASE INPUT THE LENGTHS OF THE SIDES
A,B,C ? 12,13,26
INCORRECT DATA: TWO SIDES OF THE TRIANGLE
NOT LONGER THAN THE THIRD
PLEASE INPUT THE LENGTHS OF THE SIDES
A,B,C ? -1,2,3
END OF SESSION
```

让我们来解释刚才所做的事情，

行1用键词REM(REMARK的缩写，注解的意思)开头，在它后面可插进注解的內容，用于解释程序是做什么的或如何做的，REM语句可用在程序的任何地方，可以用LIST将它显示出来，但在程序执行过程，计算机却不理睬它，它是写给程序的读者，而不是给计算机的。

行10类似于前面所用的INPUT语句，只是多了一时可插入信息的引号，该信息在提示你输入数据的问号“?”前被打印出来。

同样，行80类仍于前面所用的PRINT语句，仅是在变量T之前加了用引号括起来的信息，並在此信息和变量之间用分号分开，这样就有可能插入用以解释T的意义的信息。

如果数据是不正确的，由行40，50和60转到行100打印出错误的信息，象输入的第二组数据就是如此，其中12+13<26。

比较一下程序3·7和程序3·6，看来前者包含更多的词，信息和解释(用作“说明性文件”)，其目的是使得程序和输出的意义更清楚和更易阅读。对一个简短的程序来说，程序的思路始终很清晰，这种“说明性文件”也许是不必要的，但对一个长程序，对初次接触程序的人，或什至你自己在搞了几星期之后，这种“说明性文件”就是很需要的了。现在你刚开始学习程序设计，很希望你形成艮好习惯，随时用“REM信息”说明你的程序。

3·8 程序设计的简单练习及答案

记住我们的座右铭："学习程序设计靠动手"，可以肯定，当你遵照这种方法並在COMX－35上做出各种试验之后，那么你对计算是怎么一回事就有了更多的了解，而且有了进一步学习的良好基础，"透彻的理解来源于实践"，下面的练习为你提供更多实践的机会。

复利计算

问题的叙述

给定，P＝本钱或期初总数(＄)

R＝年利率(％)

Y＝年限

T＝一年中计利的次数，例如，

若每天计算，则T＝365

要求编写一个程序输出

F＝本利和，或期终总数(＄)，

建议

看答案之前先编自己的程序，学习用另外的方法。

解答

```
10    REM THIS PROGRAM COMPUTES THE
      PRINCIPAL PLUS
20    REM     INTEREST F, GIVEN
30    REM     P=THE PRINCIPAL($)
40    REM     R=ANNUAL INTEREST RATE(%)
50    REM     Y=NUMBER OF YEARS
60    REM     T=NUMBER OF TIME/YEAR INTEREST
              IS COMPOUNDED
70    REM     INPUT P=-999 TO STOP
75    PRINT   "THIS PROGRAM COMPUTES
              PRINCIPAL PLUS COMPOUND INTEREST"
```

```
77    PRINT
78    PRINT
80    INPUT  "PRINCIPAL" P
85    IF P=-999 THEN GOTO 210
90    INPUT "ANNUAL INTEREST" R
100   INPUT "NUMBER OF YEARS" Y
110   INPUT "NUMBER OF TIMES INTEREST IS
             COMPOUNDED IN A YEAR" T
115   REM    LINES 120 TO 150 DETECT ERRORS
             IN THE INPUT DATA
120   IF P<0 THEN GOTO 190
130   IF R<0 THEN GOTO 190
140   IF Y<0 THEN GOTO 190
150   IF T<1 THEN GOTO 190
160   F=P*(1+R/(100*T))↑(Y*T)
165   PRINT
170   PRINT  "PRINCIPAL PLUS INTEREST=$"; F
180   GOTO   77
190   PRINT
195   PRINT  "INCORRECT DATA TRY AGAIN"
200   GOTO   77
210   PRINT
215   PRINT  "END OF SESSION": END

RUN

THIS PROGRAM COMPUTES PRINCIPAL PLUS
COMPOUND INTEREST.

PRINCIPAL?  1000
ANNUAL INTEREST?  15
NUMBER OF YEARS?  5
NUMBER OF TIMES INTEREST IS COMPOUNDED IN
A YEAR?  365

PRINCIPAL PLUS INTEREST=$2115.99
```

```
PRINCIPAL?  15000
ANNUAL INTEREST?  10
NUMBER OF YEARS?  3
NUMBER OF TIMES INTEREST IS COMPOUNDED IN
A YEAR?  12

INCORRECT DATA TRY AGAIN

PRINCIPAL?  15000
ANNUAL INTEREST?  10
NUMBER OF YEARS?  3
NUMBER OF TIMES INTEREST IS COMPOUNDED IN
A YEAR?  12
PRINCIPAL PLUS INTEREST=$20222.6
PRINCIPAL? -999

END OF SESSION
```

评价解答

你自己编的程序不会与所给的解答完全相同，主要有些什么不同，想想那一个更好，如果是你的好也不要惊奇！

碳元素测年法

一件古代文物，例如找到一块化石或一具木乃伊，往往对确定它的年代感兴趣，其中一种方法就是碳元素测年法。

COMX－35附注：关于碳元素测年法

通过科学测定物体中所含碳14（一种放射性同位素）与大气中碳14含量的百分比P，再由放射性的衰变率R，就可用下式确定物体的年代T：

$$\frac{P}{100}=e^{-RT}\ \cdots\cdots\cdots\cdots\cdots\cdots\cdots(1)$$

其中R由半衰期决定，所谓半衰期是指碳同位素失去一半放射

作用所需的年限，因此

$$R = \log e^2 / H \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots (2)$$

由〈1〉和〈2〉得

$$T = -H\log\left(\frac{P}{100}\right)/\log e^2$$

$$= H\mathrm{Abs}\left(\log\frac{P}{100}\right)/\log e^2$$

其中H等于5730年，故给出P，就能计算T。

问题的叙述

给出P，即物体中"碳14"的百分数，科学上可用下式计算物体的年代T（以年计）

$$T = 5730\mathrm{Abs}\left(\log\frac{P}{100}\right)/\log e^2$$

要求你编写一个附有适当说明性文件的程序，它能接受用戶输入口，输出物体的年代T。

解答

```
10   REM     THIS PROGRAM DETERMINES THE AGE OF
20   REM            AN OBJECT BY CARBON DATING
30   REM     P=THE PERCENTAGE OF
40   REM         CARBON 14 IN THE OBJECT
50   REM         RELATIVE TO THAT FOUND IN
60   REM         THE ATMOSPHERE
70   REM     T=AGE OF OBJECT IN YEARS
80   REM     TYPE P=999 TO STOP
90   PRINT   "CARBON DATING"
100  PRINT
```

```
105 INPUT  "PERCENTAGE LEVEL OF RADIO-
            ACTIVITY" P
110 IF P=-999 THEN GOTO 180
120 IF P<0    THEN GOTO 160
125 IF P>100  THEN GOTO 160
130 T=5730*ABS(LOG(P/100))/LOG(2)
140 PRINT  "AGE OF OBJECT IN YEARS=";T
150 GOTO   100
160 PRINT  "INCORRECT DATA : TRY AGAIN"
170 GOTO   100
180 PRINT  "END OF SESSION" : END

RUN

CARBON DATING

PERCENTAGE LEVEL OF RADIOACTIVITY ? 52
AGE OF OBJECT IN YEARS=5405.78
PERCENTAGE LEVEL OF RADIOACTIVITY ? 120
INCORRECT DATA : TRY AGAIN
PERCENTAGE LEVEL OF RADIOACTIVITY ? 80
AGE OF OBJECT IN YEARS=1844.65
PERCENTAGE LEVEL OF RADIOACTIVITY ? -999
END OF SESSION
```

3·9 进行循环（使用FOR/NEXT循环语句）

你大概还记得3·4节中那个三行程序是怎样计算世界上所有三角形的面积的！其诀窍就是使用GOTO语句转回到程序的开头，有另一种方法告诉计算机反复执行某一组指令，这样的一组指令就称为循环，我们说绕循环（即一组指令）n次或简单就说循环n次。

试；下面程序：

```
10    FOR I=1 TO 100 STEP 1
20    PRINT "COMX-35";
30    PRINT "IS CLEVER"
40    NEXT I
RUN
```

看会出现什么？ 屏上将充满一行：信息"COMX－35 IS CLEVER"，刚好出现100次。

10行中的FOR语句和40行中的NEXT语句是作为一个"FOR／NEXT"时起作用的，这一对就告诉COMX－35重复执行在其中间（行20和行30）的指令100次，即从I=1开始用步长1一直到I=100。

再试：

```
10    FOR I= 0  TO 2000 STEP 2
20    PRINT I ;
30    NEXT I
RUN
```

屏上充满"024…2000"本例中步长是2，因此输出1001个数，逐行分析如下：

表3·9 逐行说明 FOR/NEXT 循环的作用

行号	作　用
10	令 I = 0
20	打印 0
30	I 坛加 2（步长值），并转回到10
10	I = 2
20	打印 2
30	I 坛加 2，转回到10
10	I = 4
20	打印 4
30	I 坛加 2，转回到10
10	I = 6
	•
	•
	•

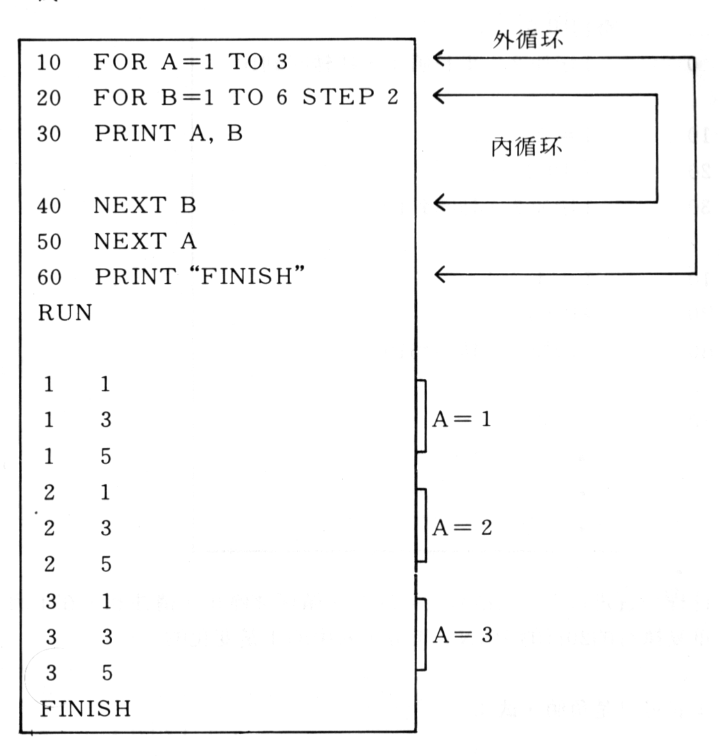
这样一直进行下去包括到 I =2000，循环才停止，请注意，在本例中，重复执行的20行的 PRNT 指令，其中的 I 是变化的。

步长可以是负值，试：

```
10        FOR A=11 TO 8 STEP−1
20        PRINT A
30        NEXT A
40        PRINT “FINISH”
RUN
11
10
 9
 8
FINISH
```

如果没有指令步长，就当作步长是1，另外，一个FOR／NEXT循环之内可以有另一个 FOR／NEXT 循环，这种情况称为循环的“嵌套”。

试



对每通过外循环一次，即对每个A值，內循环就要重复三次，顺着，上述程序，注意其中A和B的值如何变化，特别是当A从1变到2时，B在B = 1 开始。

做为最后一例，试：

```
10   FOR A=1 TO 10
20   IF A=4 THEN A=9
30   PRINT A
40   NEXT A
50   PRINT "FINISH"

RUN
 1
 2
 3
 9
10
FINISH
```

请注意行20的 IF 语句如何将循环次数从10减为 5 。

3·10 将复杂程序分块(使用子程序)——GOSUB和RETURN

设想你要编写一个能做类似于“空间入侵者”的游戏的程序，很多人由于能写这种游戏程序而致富和出名，借助聪明的COMX－35 BASIS，彩色绘图和音响效果，你也有可能做到这一点，　。

一种方法是鉴别各次要重复执行的作业，例如作业 1 是在屏上任意位置驱动空间飞船，作业 2 是驱动火箭发射器，作业 3 是当空间飞船被击中时产生彩色图象和音响效果，作业 4 是当火箭发射器被击中时产生彩色图象和音响效果，作业 5 是断定空间飞船是否已被击中，等等。

为了模拟一艘运动中的空间飞船，在原始位置驱动空间飞船(用所谓作业 1)，但使用背景的颜色，这样将擦掉原始的空间飞船，现在将空间飞船从其原来位置沿飞行方向稍为移动，如果这样迅速地重复许多次，就得到一艘在飞行的空间飞船，但这时就要求对作业 1 作用很多次。

现在，作业1用一程序块来实现，如果凡是在空间飞船被驱动的地方都插进这个程序块，那么整个程序就需要很长，另一种方法是将作业1的程序（称为子程序）储存在某个地方，然后在主程序中处是空间飞船被驱动的地方就插入"转子程序"即"GOSUB"语句，图3·10对此做了说明，其中带号码的箭头相应于下列各步。

第一步　当计算机遇到行110的"GOSUB 5000"时，就转移控制到行5000的子程序的开头。

第二步　执行子程序中的指令，一直到最后一条指令RETURN。

第三步　控制返回到行120，即紧接在行110的"GOSUB"语句之后的那一行。

第四步　继续执行主程序

第五步　计算机在行210遇到另一个"GOSUB 5000"，象第一步一样，控制转移到子程序的开头。

第六步　同第二步。

第七步　控制返回到行220。

第八步　继续执行主程序。

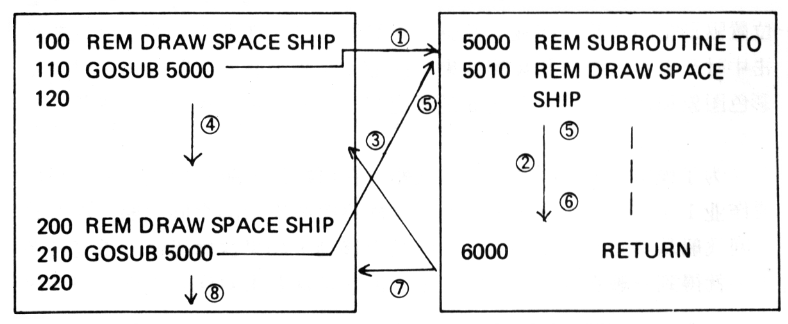


图3·10 分步说明主程序用GOSUB调用子程序的动作。

注意，在“GOSUB N”中，N是子程序的头条指令的行号，而且，子程序总是以RETURN语句结尾。

为了说明子程序的用法，下面编一个找出三个整数A，B和C的最大公约数即GCD（也称为最高公因子，HCF）的程序，我们需要一个找二个数的GCD的子程序。设A，B的GCD是Y，然后再调用子程序找Y和C的GCD，那么所得结果就是A，B和C的GCD。

最大公约数

```
10   REM     THIS MAIN PROGRAM
20   REM     THE GCD OF INTEGERS
30   REM     A,B,C
40   REM
42   PRINT   "THIS PROGRAM FINDS THE GCD"
43   PRINT
50   INPUT   "A", A
55   IF A<0 THEN GOTO 150
60   INPUT   "B", B
70   INPUT   "C", C
80   X=A
90   Y=B
100  GOSUB   1000
110  X=C
120  GOSUB   1000
130  PRINT   "THE GREATEST COMMON DIVISOR="; Y
140  GOTO    50
150  PRINT   "END OF SESSION"; END
1000 REM     THIS SUBROUTINE FINDS  THE GCD OF
1010 REM     TWO INTEGERS X AND Y
1020 REM     AND THE GCD=F
1030 REM
1040         Q=INT(X/Y)
```

```
1045        R=X-QY
1050        IF R=O THEN GOTO 1100
1060        X=Y
1070        Y=R
1080        GOTO 1040
1100        RETURN
```

3·11 进一步编辑的方法

由于编写较长的程序时，很可能常发生错误，因此用2·1节和3·3节所介绍的编辑方法可能就不夠适用，编辑时除了使用“DEL”键，“CNTL”和“行号”（见2·1节和3.3节，实际上COMX－35还提供更實用的方法来修正程序中的错误。

这可通过使用命令EDIT来实现，其形式是

EDIT X

其中X是整数，记需要修正的语句的行号。

机器对EDIT X的应答是显示出行X，这时计算机是在“编辑状态”（即在EDITOR的控制下而不是在“COMX－35 BASIC”状态，后者使COMX－35处于BASIC解释的控制下（见4·3节）），用戶可移动光标（按空棒）到所显示的下面需要修改的字符之前（即其右边）一个字符的位置上，这时有三种可供选择的编辑方法，用戶可打I（用于插入），D（用于删去），或C（用于更换），挑选好那一种选择，相应就显示出所选择的字符（I，D或C），同时光标即落到要修改的字符下面。

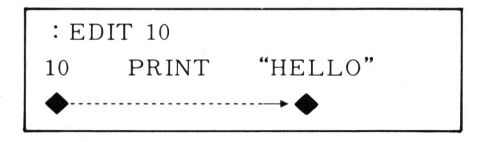
选择插入，就可将字符加进或插入到由I所指定的位置的后面，选择删去时，则每按一次空棒就依次将一个字符以行中删掉，也就是说，按几次空棒，就删去几个字符，选择更换时，就以所打入的字符依次替换C后的每个字符。

用戶在做了要做的修改之后，可打CONTROL S（按住 CNTL 键再打S），让已修改过的行X重打出来，请注意，每次只允许一种选择（I，D或C）不过，需要的话，用戶可对每种选择根据需要及复做各種改正的练习，做了练习之后，再按CNTL S 以便以编辑状态“退出”，说明对行X的编辑已完成。

上面所讲的可解释如下，其中用“u”来记空棒，打

```
10      PRINT    "HELLO"
```

为了编辑行10，打 EDIT 10 可得



其中要编辑的是行10，现在移光标到”号的下面准备接受下一步动作。

如果要在行10中HELLO之前插入SAY，则应选择I，並打 SAY 再按空棒，得到

```
:EDIT 10
10      PRINT    "HELLO"
                  ISAYU
```

现在打CNTL S，得到

```
10      PRINT    "SAY HELLO"
```

再打一次 CNTL S 以从编辑状态退出。

如果要刪去SAY，並将HELLO改为JERRY，就选择D，接着按 4次空棒，得到

```
:EDIT 10
10    PRINT    "SAY HELLO"
               Duuuu
```

注意，相应D后面的每一个u，就删去一个字符，按CNTL S显示改正过的这一行，既然编辑还未完成，就不要在这时退出编辑状态，继续下去，並移光标到被改过的行中"号的下面，同时选择更换

```
10    PRINT    "HELLO"
               CJERRY
```

按 CNTL S 就显示出

```
10    PRINT    "JERRY"
```

注意，5个字符HELLO已被另外5个字符JERRY所替换處现在编辑工作已完成，再按CNTL S退出编辑状态，计算机将应答以

```
READY
:◆
```

此即说明控制已返回到COMX－35 BASIC解释状态。

若在一个简短的行中出现错误，则编辑的最简单方法是用同一行号重打这整行，对于一个较长的行，在修改老的错误时有可能出现新的错误，在这种情况下，最好用EDIT。

在编辑状态下按CNTL C将会使编辑状态失效幷退回到BASIC状态而留下要编辑的行不变。

3·12 产生特定的图示字符

3·12·1 內部“图示字符”——用函数 CHR $

要试机內的“图示字符”，可执行下面的程序

程序 3·12·1

```
10      FOR I=0  TO 127
20      PR I, CHR $(I);"      ";
25      IF I=127  THEN PR
30      NEXT

RUN
```

关于函数CHR $（X）的应用，在5·5 节中有详细介绍，简要而言，CHR $（X）是一个函数，它对每个给定的整数X，给出一个字符，对每个键，相应地都有一个整数或代码（严格地说是ASCII代码，其中ASCII是“Ameriean Standard Code for Information Interchange”的缩写）和一个字符，例如键A的代码是65（65是十进数，写成16进制就是41）通常按下键A时就输出字符A，因此“PR CHR $（65）”就等价于“PR A”。所以上述程序将显示出代码是0到12 7所相应的全部128 个字符，现在，当COMX－35 第一次运行时，相应于十进代码32到95和160 到223 的字符给出的是标准字符（即是S P！等等包括1, 2, ……9, A,B,C,……，Z 等）为附录A·2所示，相应于十进代码 0 到31，96到127，144 到159 和224 到255 的键给出內部图示字符，例如PR CHR $(21)将打印出青色的六边形。

试打

```
PR    CHR $    (111,112,113)
```

並按ＣＲ，这时将打印出三个内部字符边连边构成一幅飞机图案。

若要得到比较长的飞机，可打

```
PR    CHR$  (111,112,112,112,113)
```

如果按住 SHIFT 键，再按Ｏ键（一次），Ｐ键（ 3 次）和Q键（ 1 次），也可得到同样的图案。

3·12·2 用戶自定文字符——使用 SHAPE 指令

可以重新定义某个键所代表的字符，例如可以对十进代码65重新定义一个字符，使得按下键Ａ时给出的不是字母Ａ而是一个汉字。

这可用指令SHAPE实现，其形式为（对使用 PAL 制的电视）：

SHAPE(X,“18位十六进数”)

其中Ｘ是整数，相应于该键的十进代码，一对双引号內的是18位十六进数，用它来指定字符的颜色和形状，试打

```
10    SHAPE(65,“48487E6A7E48484800”)
RUN
```

（注意：SHAPE 也可在不带行号的直接执行方式中使用。）

这时按键Ａ就会印出粉红色的汉字“中”，事实上，我们要到，在执行上述SHAPE指令之后，屏上所有的字母Ａ的地方都代以中字。

再试试下列一些例子，你就能够领略这些指令的奇妙的魔力。

例 1　SHAPE(153,“8F8D8C5C7E7E7E5C00“)

当你打“PRINT CHR$(153)”时就显示出一个苹果。

例 2　SHAPE(119,“84848CBF9E9C94A400”)

只要你按按下 SHIFT 键时按字母“W”，计算機就相应给出一颗绿色的星。

例 3　SHAPE(20,“ 00000000FFFFFDF00”)
SHAPE(21,“00FCFCFCFFFFFFFF00”)
SHAPE(22,“00000000FEFFFFFE00”)

在按CNTL－T，CNTL－U和CNTL－V之后，就能在屏上得到一艘潜水艇。

对使用NTSC制的电视，不用18位十六进数，而只用16位十六进数来指定其形状。

COMX－35 关于十六进数的附注

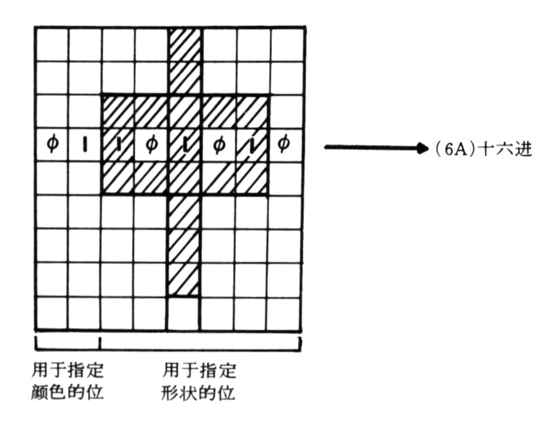
十六进就需要16个不同的字符，因此，除了0,1,… 9 外，还需要 6 个字符来代表“10”,“11”,…“15”，通常就取A,B,C,D,E,F ，使用 4 位一组的方法可将十六进数与二进数互化，因此，

111/1010/0001/0010/0001

变成十六进数就是　7A121

为了确定此 9 对十六进数，考虑一个 8 单位× 9 单位的矩形如下：

图 3·12 为得到字符“中”如何写 SHAPE 指令



考虑每行的8个方格，最左边二个方格，可填1和0，按照表3·12·2用于决定余下的六个方格的颜色，每行最右边的六个方格的每一个如果该方格要“被打印”或“被填满”就相应取1，否则（即是说，如果它与背景颜色相同）就取0。因此，头二列方格（从左算起）用于决定颜色，而最右边6列方格（6×9个方格）用于决定形状，每一行编成二位十六进数代码9行，一共就需要18位十六进数来同时决定颜色和形状。例如，图3·12中从顶上数起第4行，如果要选择粉红色，那么最左边二位应是“01”（见表3·12·2）剩下6位为图312所示应是“101010”，“填满方格”的代码是“1”，“留空方格”的代码是“0”，这个八位式“0110 1010”可代以2位十六进数6A，因此，第4对（相应于第4行）十六进数是6A（见图3·12），要注意，对PAL制，每个字符的分辩力是6×9个点，对NTSC制，只需8对十六进数来给出6×8个点，如果位数比需要的多，则余下的机器不予理睬，对这两种情况每个字符都可以是多种颜色，还要注意，当定义了形状之后，可用字符串变量代替文字，而且用戶重新定义的字符或符号可像其他原有的字符一样打印出来。

这种重新定义的字符在切断电源或系统重置（即同时按 RT 键和空棒）时就复原。

表3·12·2 每行方格的颜色当作头二位的函数

最左边二位	颜色（如果用计算机输出）	颜色（如果从键盘输入）
0　0	黑色	红　色
0　1	蓝色	粉红色
1　0	绿色	黄　色
1　1	青色	白　色

3·13 时间控制的子程序——指令 TIMOUT 和 TIME 的用法

联合使用指令 TIMOUT 和 TIME 就使程序员能在经过指定的时间后调用子程序，使用指令

TIME(X)

其中X是整数，就可做成一个“闹钟”，当执行上述指令时，“时钟”就开始滴嗒响，过了X单位时间之后，指令

TIMOUT Y

起作用，它指挥计算机跳转到其起始引号为Y的子程序

在COMX－35的PAL制中，50个单位相当于1秒。

在COMX－35的NTSC制中，60个单位相当于1秒。

含有时间控制子程序的程序的典型结构及其执行的顺序在图3·13中加以说明

第1步：遇到指令TIMOUT（1000）计算机记住子程序的开始地址，但仍未有其他动作，

执行指令 TIME(150)“钟”开始滴嗒响。

第2步：计算机照常执行。

第3步：在执行语句100之后再经过150个时间单位（对PAL机即是3秒，对NTSC机是2.5秒），“闹钟”就指挥计算机跳转到子程序的开头。

第4步：执行子程序，通常在子程序里面也放一个“钟”，但也不是一定的，语句1990说明该“钟”是用以在经过100个时间单位后发出“警报”的。

第 5 步：执行到RETURN，控制返回到TIMOUT后面的语句20。

第 6 步：计算机照常执行下去，一直到从语句1990算起，经过 100 个时间单位时，控制再度转向子程序。

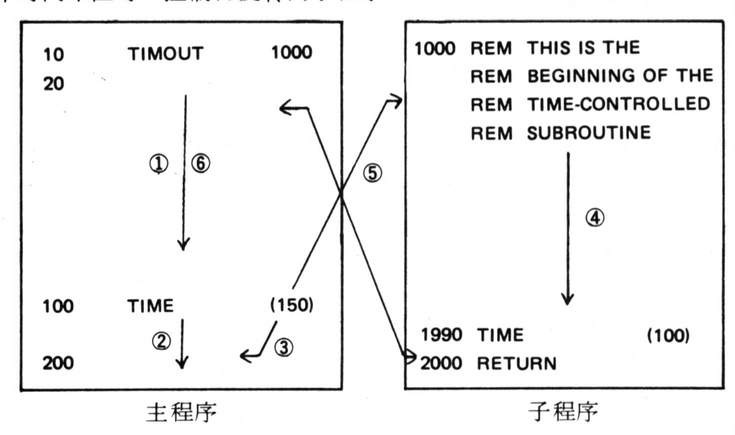


图3·13 使用时间控制子程序的程序的典型结构和执行顺序

3·14 控制（CNTL）键的功能

若在按下控制（CNTL）键时按某个键，计算机会做某些动作，但屏上并不显示字符。

CNTL C

正如 2·1 节所说，如果在按住CNTL键时按键C（这个操作简记为"CNTL C"）再按C R 键，计算机将不理睬正在打的行，同时光标则跳到下一行的最左边位置，CNTL C使得用戶能够命令计算机不理睬部分打过的语句。

CNTL R

CNTL R命令计算机将最后一个语句重新在屏上显示，当要打入的新语句与最后语句有些不同时这个命令就特别有用，这时可用D E L键和重打的办法来修改最后语句，然后按C R 键将新语句送进计算机。

移动光标（CNTL，I，K，M，J）

当按住CNTL键时按键I，K，M和J就可移动光标的位置如下：

I：光标移上一行（↑）
M：光标移下一行（↓）
K：光标往右移一列（→）
J：光标往右移一列（←）

注意，当直接从键盘上执行的，移动总数不能超过95。

在程序中用PRINT CHR＄(128)来使光标往上移
在程序中用PRINT CHR＄(130)来使光标往下移
在程序中用PRINT CHR＄(129)来使光标往右移
在程序中用PRINT CHR＄(131)来使光标往左移

3·15 程序的实时控制——使用KEY的操纵杆

KEY

这个函数是给出被按的键的ASCII代码，其用法可用下列说明

```
10    IF KEY=65 THEN GOTO 30
20    GOTO 10
30    PRINT "KEY A PRESSED"
40    END
```

当遇到函数KEY时，就检查键盘，将所按的键转换成其ASCII代码，如果它等于65（那是键A的ASCII十进代码），则打印出信息“KEY A PRESSED”同时程序终了，如果按下的不是键A，则程序进入循环，反复检查键盘。

上述程序说明使用函数KEY的典型情况，它常常用在循环中的“IF语句”里面，使得机器反复扫描键盘，如果某指定键被按下时就做某次动作，这样就可以通过按下指定键来控制程序的流向。

操纵杆

机上的操纵杆是用来控制 4 个键的，按照操纵杆所按的方向，将 4 个代码中的一个输入计算机，这 4 个代码就是 ASCII 十进代码 136, 137, 138和139，分别对应于操纵杆向上，向右，向下和向左的位置，如果语句“ IF KEY=136 THEN…… ”出现在循环的某个地方，那么只要操纵杆被拨到向上位置（它的ASCII十进代码是136），IF语句的条件就是被定，这样就可用操纵杆来控制程序的流向。

试：

```
10    IF KEY=136 THEN GOTO 30
20    GOTO 10
30    PRINT "JOYSTICK IS ACTIVATED"
40    END
```

当操纵杆被拨到向上的位置时，程序就打印出信息。

第四章

计算机基本概念

第四章　计算机基本概念

4·1 计算机是什么？

计算机是能够接受、处理和输出信息的设备，信息是以电子信号的形式，它代表字母式数字数据（即是文件，字和数）处理包括数学的和逻辑的运算，分类，检索和重新排列，输出“处理过的”信息主要有三种应用，（a）数据处理（包括商业和会计应用，档案保存，情报检索，管理信息系统以便更容易作出决断）（b）科学计算和工程设计，（c）工业控制，通过拾取输入信息，再用感应器或换能器转变为电信号，并以指定的方式进行处理，然后用输出信号指挥各种电气的或机械的设备动作。

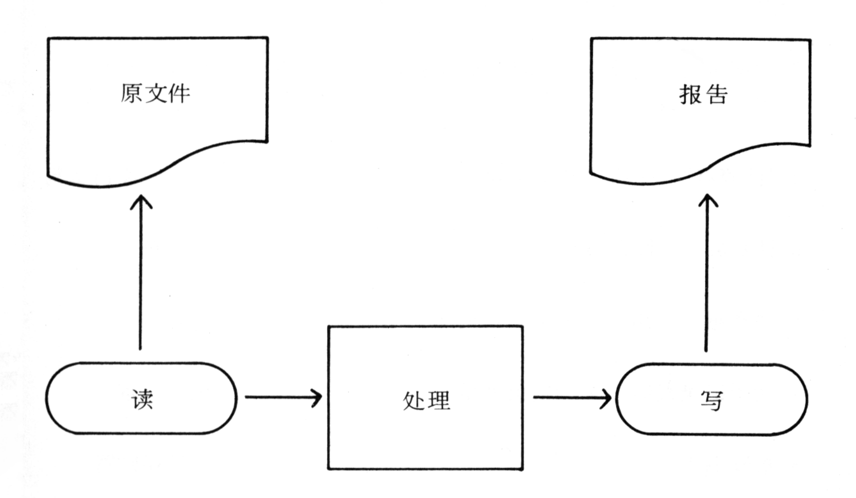


图4·1 做为信息处理机的计算机

4·2 计算机的主要部件

解答问题可以用计算机，也可以由配备有计算器，笔和纸的计算员，，比较一下这两种性况下是如何做的以便鉴别计算机的各个子系统显然是有益的，表4·2就是这种比较：

表4·2 由计算员和用计算机解答问题——其必需品的比较

计算员	计算机
1. 问题清单——用于准细记录问题。	输入设备
2. 算法：——规定解答问题过程的一系列步骤。	程序存贮器
3. 数据，按上述算法对它进行处理从而得出所需的答案。	数据存贮器
4. "计算器"——实现基本运算。	算术或逻辑单元
5. 计算员——控制算法的执行。	控制单元
6. 笔和纸——用于记录中间结果。	暂贮器
7. 用于记录最后结果的结果清单。	输出设备
8. 参攷书库。	海量存贮器

表4·2说明，计算机的组成是：

（a）用于做算术或逻辑运算的子系统和用于控制每一步的顺序的子系统，前者称为算术——逻辑单元即ALU，后者称为控制单元，在计算机中，这两个子系统常常组合在一起构成所谓中央处理单元即CPU、CPU就好像是配备有计算器设计算员。

（b）用于存贮程序，输入数据和中间结果的设备，在计算机中，这些都存贮在主存贮器中，主存贮器就好像计算员使用的纸和笔。

（c）用于计算机与外齐通信联络的输入输出设备，使计算机能接受问题的陈述，有关的数据（例如从键盘输入）和输出结果，（例如通过电视屏幕）

（d）用于存贮许许多多不太常用的信息的设备，这样的海量存贮设备（例如软磁盘或磁带）其存取速度不及主存贮器，但是能夠存贮大量的信息，它就像计算员时不时要查询的参考书库。

所以，计算机主要的系统是：ＣＰＵ、主存贮器、输入输出设备和海量存贮器。

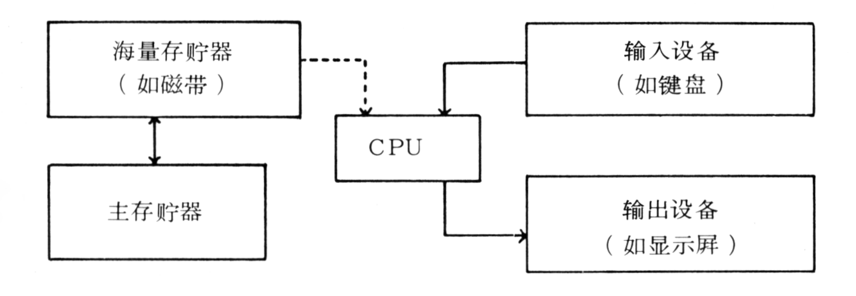


图4·2 计算机主要的子系统

4·3 计算机软件

所谓软件，我们理解为程序或能告诉计算机为何实现不同任务的一步步过程，软件主要有三类：

（a）应用软件：用于解决某些科学或数据处理的问题，例如做一般总帐、仓库管理、或用于建筑的结构设计，这些由用戶提供(或编写)。

（b）系统软件，它使用戶更便于操作计算机，例如能实现各种任务，像使用戶能将程序存入磁带或反过来回输给存贮器的程序。

（c）语言编译或解释程序，它使用戶能用近似于普通语言的语言来编写应用程序，在计算机中，程序和数据是用许多只有0和1两种“状态”的部件，（就像开关，不是开就是关，来表示的，计算机能直接识别这些“0－1”模式。然而，对用戶来说，仅用0和1来写程序是极其困难的，什至几乎是不可能的，而且写出来的程序也不容易认读，可以采用一种类似于普通语言的语言，如BASIC来描述问题的解答过程，然后交由计算机执行之前，将其转换成“0－1”（或：进行）模式。这样的转换就由称为“编译程序”或“解释程序”的软件来完成，编译是将用方级语言编写的程序整个地译成机器语言程序，然后再执行、解释是对原程序的语句转译並执行。

第五章

COMX—35 BASIC 参攷指南

第五章　COMX－35 BASIC 参考指南

本章可作为参考手册使用，它全面介绍了COMX－35 BASIC语言，不熟悉 BASIC 和第一次阅读本章的用戶可能发觉它枯燥无味 难以读下去，但随着你更加熟悉 BASIC 之后，就会发现这一章是知识的不可估量的源泉。

5·1 程序、语句、表达式和函数

程序是计算机能执行的语句的集合，这些语句告诉计算机如何解答问题，它将整个方法具体表示成称为“算法”的一步了，过程的形式和做为算法运算对象的数据。

语句，（用引号开头）由 BASIC 键词（如END，NEXT）表达式和函数组成。

函数是给出数值的 BASIC 键词，例如LOG(X)，SIN(X)等在 5·4 节中给出一个函数表，据弧內的量称为函数的自变量。此键词是规定某个规则，据此可将自变量转換成某个值，后者称为函数值。

表达式是算术表达式的简称，它可以是单个数、单个变量，函数或由算术算符连接起来的一群操作数（数、变量和函数等）例如：8 、5 、A 、3 * Sin(45) 0.5　3 *（ 2 + A)/(B + C)

5·2 数和变量

COMX － 35 既处理整数也处理浮点数，两种数都以32位 2 进制的有符号数的形式存贮，浮点数的数值范围是从 -0.170141×10^{39} 到 $+0.170141\times10^{39}$，尾数的小数部分精确到 6 位数字。

整数的范围是从－2147483647到＋2147483674在这范围內都是精确数。

从键盘输入的数都以所输入的形式存贮起来，如果输入時沒有小数点，则做为整数储存，如果是带小数点或有“E”记号，则做为浮点储存。

5·3 算符

COMX — 35 的算符可分成五类：算术、赋值、比较、逻辑和其他。

算术算符

下表给出算术算符的符号和它在数学表达式中运算的优先顺序：

一元减：	—	第一优先级
指　数：	↑	第一优先级
乘　：	*	第二优先级
除　：	／	第二优先级
加　：	＋	第三优先级
减　：	—	第三优先级

同一优先级的算符按从左列到右的顺序运算，符号“—”也用来记负数，下面是一些说明优先级的练习。

4 * 3 ↑ 2 ＝36(先做 3 ↑ 2 ，再做 4 *(3 ↑ 2))
4/2 * 3 ＝ 6
4 ＋ 2 * 3 — 5 ＝ 5

赋值算符

赋值符号就是等号“＝”将这个符号用于赋值时，其意义是“取什么为值”而不是“等于”。

例如：
```
LET  A=5
LET  B=B+A* 2
```

键词LET可任意省略，第二个例子可解释为：计算B＋A * 2的值並将结果送给B 。

POKE（表达式１、表达式２）

这个语句使用时应特别谨慎，本语句是用来将某个数据写进存贮器的某个位置。存贮器的位置由表达式１所指定，而数据由表达式２决定，例如POKE（5000,255）就是将255的十六进数（ＦＦ）放到十进制地址5000的存贮单之中，通常更稳妥的办法是直接写为POKE(■1F23,＃ＦＦ）它将十六进数ＦＦ放到十六进地址1F23的地方。

关系算符

关系算符包括：

等　于：＝
不等于：＜＞
大　于：＞
小　于：＜
大于或等于：＞＝
小于或等于：＜＝

包含关系算符的表达式，其计算的结果是眞或假，而不是数值，例如：

```
IF A＝３＋B＞２ THEN
IF A ＜＞４ THEN
IF A＊２＜＝B／４ THEN
```

逻辑算符

逻辑算符是AND,OR,XOR NOT。应该注意，在做指定运算之前，逻辑算符将表达式转换成整数，另外，在ＮＯＴ算符后面的表达式必须放在括号內。逻辑算符与＋和－属同一优先级，例如：

```
IF（A AND B）＜＞C THEN
FOR I=（A AND B）TO（A OR B）STEP A/B
PR（A XOR B）
GOTO NOT（A）
```

头三个例子中，为清楚起见都加了括号，其实都可省略。

其他算符

其他算符罗列如下：

：	分隔同一行中的语句
；	PRINT 的定界符
，	PRINT 的定界符
”	字符串的定界符
（	组限界
）	组限界
\...\	8 位二进制值
#	8 位十六进制值
■	16位十六进制值

下面是一些例子：

```
PRINT A,B :"YES" : GOTO 10
LET A=( 3 + 4 )/2
LET B=#F8:C=■01F3
IF A<>\10011010\THEN GOTO 100
```

注意 1 ，在 PRINT 语句中使用；时，打印出来的下一项紧接在前一项后面，中间不留任何空位。

注意 2 ，在 PRINT 语句中使用，时，打印出来的下一项是位于下一个打印区的开头位置，一个打印区的宽度是 8 个字符。

注意 3 ，使用 ” 是指字符串的界限，因此不能作为字符串內的符号使用。

5·4 数学函数

在第一小段给出专门函数的字义，数学函数的自变量可以是数，变量或表达式，函数可分成(i)內部函数和(ii)导出函数，所谓导出函数是可以由內部函数的表达式得到的函数，例如SIN(X)和COS(X)是內部函数，而TAN(X)是导出函数，因为TAN(X)可由SIN(X)/COS(X)计算。

COMX－35的预置状态是假定用弧度来表示角，下面的例子都假定是这种状态，若要改变成角度，可打DEG接着再按CR键，要再转回改弧度，就打RAD再按CR键。

5·4·1 內部函数

內部函数包括ABS, ATN ,COS, EXP, FNUM, INT, INUM, LOG, MOD, PI, RND, SGN, SIN, 和SQR。

ABS(表达式)

这个函数给出此表达式的绝对值

例如　PR　ABS(－10＊2)

将打印出20

ATN(表达式)

这个函数与ARCTAN(表达式)同样意思，即给出一个其正切等于此表达式的值的角(通常用弧度表示)这个函数是浮点函数。

例如　PR ATN(1)

将打印出　0.7854

因为TAN(0.7854)＝TAN PI/4＝TAN45°＝1

COS(表达式)

这个函数给出由表达式的值规定的角(通常用弧度表示)的余弦，本函数是浮点函数。

例如　PR　COS(PI/3)
将打印出　0.5
因为COS　60° =0.5

EXP（表达式）

这个函数给出2.71828（ e ）的乘幂的浮点数，其指数等于此表达式的值。

例如　PR　EXP(6/3)
将打印出　7.3891
因为e 6/3=e^2 =2.71828^2 =7.3891

FNUM（表达式）

这个函数将表达式舍入到最接近的整数应将它转为浮点方式，它与INUM刚好是相对。

例如　PR　FNUM（RND(6)）
将打印出浮点值

INT（表达式）

这个函数将浮点表达式的小数部分分削去而给出其整数部分，其结果仍旧是浮点方式，不要将这个函数与（下面要说明的）函数INUM相混淆。

例如　A=7.9：　B =INT(A)：　PR A, B
将打印出　7.9　7

INUM（表达式）

这个函数将浮点表达式转换为整型，取其最接近的整数，它提供一种把特定函数变为整型的方法，

例如　PR　SQR(62.41)　结果是7.9
而　　PR　INUM(SQR(62.41))　结果是 8

LOG（表达式）

这个函数给出表达式的值的自然对数值的浮点数。

例如　PR　LOG(10)
将打印出　2.3026
而　　PR LOG(EXP(2))　将得到 2

MOD（表达式 1 ，表达式）

这个函数（Modulo）等价于下式的值：

表达式 1 —（表达式1/表达式 2 ）*（表达式 2 ）其中每个表达式都要先化为整型，这个函数给出一个整型值，

例如　PR MOD(10,3)
其结果为 1

PI

这个函数给出数值3.14159

RND

这个函数给出一个大于等于 0 ，小于等于 1 的随机浮点数

RND（表达式）

这个函数给出一个大于等于 0 ，小于表达式的值的随机整数。

如果要将所产生的随机整数赋给一个变量，那么此变量必须先规定（为整数值型的（例如：DEFINT A）

SGN（表达式）

这个函数将根据表达式的符号分别取值+1，0或-1。如果表达式为正，则函数给出+1，如果表达式的值为0，则函数给出0值，如果表达式为负，则函数给出-1。

SIN（表达式）

这个函数给出由表达式的值所给定的角（通常用弧度表示）的正弦，这个函数是浮点函数，

例如　PR　SIN（P1/6）
将打印出0.5，

SQR（表达式）

这个函数给出表达式的值的平方根，本函数也是浮点函数，

例如　PR　SQR（65）
将打印出8.0623

5·4·2 关于数学函数的一些例子

下面一些例子可说明COMX-35 BASIC数学函数的功能，函数的自变量本身可以是函数或表达式，这样就能夠表达相当复杂的数学式子，

```
PR SIN（45）
A=INT（10*SQR（LOG（A*B）））
C=EXP（10+A）
PR ATN（A*SIN(A)）
```

5·4·3 导出函数

下列函数不属COMX-35 BASIC的内部函数，但可由5·4·1即所列的内部函数用下列公式计算，

```
TAN(X)=SIN(X)/COS(X)
ARC SIN(X)=ATN(X/SQR(1-X*X))
ARC COS(X)=-ATN(X/SQR(1-X*X))+1.5708
ARC SEC(X)=ATN(SQR(X*X-1)+(SGN(X)-1)*1.5708
ARC COSC(X)=ATN(1/SQR(X*X-1)+(SGN(X)-1)
  *1.5708
ARC COT(X)=-ATN(X)+1.5708
ARC SINH(X)=LOG(X+SQR(X*X+1))
ARC COSH(X)=LOG(X+SQR(X*X-1))
ARC TANH(X)=LOG((1+X)/(1-X))/2
ARC SECH(X)=LOG((SQR(1-X*X)+1)/X)
ARC CSCH(X)=LOG((SGN(X)*SQR(X*X+1)+1)1X)
ARC COTH(X)=LOG((X+1)/(X-1))/2
COT(X )=1/TAN(X)
CSC(X)=1/SIN(X)
SEC(X)=1/COS(X)
COSH(X)=(EXP(X)+EXP(-X))/2
COTH(X)=EXP(-X)/(EXP(X)-EXP(-X))*2+1
CSCH(X)=2/(EXP(X)-EXP( -X))
SECH(X)=2/(EXP(X)+EXP(-X))
SINH(X)=(EXP(X)-EXP(-X))/2
TANH(X)=-EXP(-X)/(EXP(X)+EXP(-X))*2+1
```

5·5 字符串和字符串函数

字符串即字符（可打印的字符或不可打印的字符）序列，字符串中可以包含空格，但不能包含引号”，因为引号是作为字符串的分界号，即用来表示字符串的开始和结尾，对COMX－35 BASIC，每一字符串可最多包含127个字符。

作为非数值数据的字符串可用作标识、信息等，巧妙地处理数值的机能力使计算机能做科学计算，同样地，巧妙地处理字符串的能力使计算算机能做字处理，档案保存和情报检索。

处理字符串的函数有：ASC, CHR＄ FVAL, LEN和MID＄。

ASC（字符串表达式）

这个函数给出该字符串的第一个字符的十进的ASCII值，该值是以浮点形式给出的，如果接在整型函数后面就变为整型的。

例如　A$(5)="ARITHMETIC"
　　　B=ASC(A$(5))

变量B的值是65，即是字母A的十进的ASCII值，

又如另一个例子

```
 5    A$="TEST"
10    FOR A=1 TO LEN(A$)
20    PR ASC(MID$(A$,A))
30    NEXT A
```

将给出

```
84
69
83
84
```

CHR$（表达式，表达式，…）

相应于括号内的每一个表达式，这个函数给出一个字符，因此，CHR$的作用与ASC正好相反，它先计算每个表达式的值并输出ASCII十进代码中等于该表达式的值的那个字符。

例如　PR CHR$(65)

将给出

```
A
```

因为A的ASCII十进代码是65，又

PR CHR $(#42)

将给出

B

因为B的ASCII十六进代码是42。

PR CHR $(#41, #42, #43)

将给出

ABC

有关使用CHR $的其他例子可参看3·12节附录A.2给出所有內部字符的ASCII代码。

FVAL（字符串表达式）

这个函数的COMX－35所特有，它将字符串表达式当作算术表达式一样进行计算並给出其值用戶就可以这种方式制造数学函数，下面是包含FVAL用法的三个例子

```
10       A $ =“8 + 4”
20       PR FVAL(A $)
30       PR FVAL(A $ +“/2”)
40       PR FVAL(“SQR( 3 )”)
:        RUN（执行程序）
12       （计算幷打印A $）
10       （计算幷打印(“8+4/2”)）
1.73205  （计算幷打印SQR( 3 )）
READY
:◆
```

规定字符串不能超过48个字符，如果超过此长度，会冻结掉有用的数据范围。

LEN（字符串变量）

这个函数给出所指定的字符串中字符的个数，它取为浮点值，除非是接在整数式整型函数的后面，这时它会自动转为整型数，下面是一个例子，

A＄＝"ARITHMETIC"
PR LEN(A＄)

给出

10　　（A＄的长度）

又

PR 3 * LEN(A＄)

给出

30

注意，字符串必须预先有定义。

MID＄（字符串变量，表达式 1 ）

MID＄（字符串变量，表达式 1 ，表达式 2 ）

这个函数是一个字符串函数，执行时，它取出所指定的字符串的一部分，头一次，即表达式 1 规定从左算起第几个字符作为子字符串的开头，表达式 2 规定子字符串中字符的个数，如果沒有表达式 2 ，则所有剩下的字符全取。

下面是一个例字

```
A$="YES"
B$=MID$(A$,2,1)
```

那么B$将包含字母"E"

```
A$(10)="EXPERIMENT"
PR MID$(A$(10),7,3)
```

将打印出

```
MEN
10      INPUT A$(1)
20      IF MID$(A$(1),1,1)="Y"GOTO 100
30      GOTO 10
100     END
```

上例要一直等到有某个以字母"Y"为开头的词输入时才跳转到100行

在其他 BASIC 版本中通用的一些函数如LEFT$和 RIGHT$ 都能用 MID$函数以下述方法来实现：

```
MID(A$,1,N)
```

就与LEFT$(A$,N)的作用相同，而

```
MID$(A$,(LEN(A$)-N+1),LEN(A$))
```

RIGHT$(A$,N)的作用相同

字符串的联接

所谓联接是将两个或多个字符串加在一起，比如"HELLO"和"JANE"相加就得到另一字符串"HELLOJANE"，做联接时，使用+（加）号並用字符串或字符串变量构成赋值语句。

关于字符串的联接的唯一限制是：由两个或多个字符串相加生成的字符串不要超过127个字符。

例如：

```
10  A$="NEVER"
20  B$="THE"
30  LEF A$(5)="LESS"
40  LET B$(1)=A$+B$+A$(5)
50  PRINT B$(1)
```

40行的结果是产生一新的字符串B $(1)，它是将前面三个字符串串接起来，在50行就打印出“NEVERTHELESS”。

输入字符串，在IF语句中使用字符串

考虑下面的例子：

```
10  INPUT B$
20  A$(1)="CONTINUE"
30  A$(2)="PHASE 1"
40  A$(3)="PHASE 2"
50  IF B$=A$(1)+A$(2) GOTO 100
60  IF B$=A$(1)+A$(3) GOTO 200
70  GOTO 10
```

上面的程序将停在10行处等待输入一字符串，该字符串就存贮在B $中，然后将它与A $(1)+A $(2)和A $(1)+A $(3)比较，按照相同与否，程序转到相应的地方执行。

5·6 数组

若干次的集合，所有的项都带有同样的组名，但每一次又由一数（或一组数）所指定，则称其为数组，例如$a_1, a_2, \cdots a_n = A[1:n]$ 是大小如n的一维数组（表或向量），数组中每一之素，为ai由数 i 所指定，它表示数组中从第一个之素算起该之素的位置。

$$A[1:m,1:n]=\begin{vmatrix} a_{11}, a_{12}, & \cdots & a_{1n} \\ a_{21}, a_{22}, & \cdots & a_{2n} \\ \cdot & & \\ \cdot & & \\ \cdot & & \\ \cdot & & \\ a_{m1}, a_{m2}, & \cdots & a_{mn} \end{vmatrix}$$

是二维数组（或表），其中每个元素有行值 i 和列值 j ，二者就规定了该元素在数组中的位置。

COMX — 35 可安置26个数组[从A（表达式）到Z（表达式）]和26个字符串或字符串数组A $到 Z $和［ A $（表达式）到 Z $（表达式）］，这26个数值数组可以是一维或二维数组、数组中每一维的最大长度是255 、最小长度是1，能容纳的最大数组是G（255，255），最小是G(1)，数组元素的最多个数受可使用的RAM的数量的限制，使用数组首先必须用DIM语句定义，这将在5·9节说明。

可以用几种方式来清除数组空间：RUN,NEW和 CLD，这些将放在关于这些语句的解释之后再来讨论，它对于知道每个数组究竟使用多少存贮单元是有好处的，数组中每个元素是 4 个字节长，而且每个数组有一个 5 个字节的信息头，因此，由DIM A（10，10）定义的数组包含100 个元素（10×10），它名字为A ，其405 个字节，类似地，用DIM B（20）定义的数组B 包含20个元素，80个字节，加上信息头总共85个字节，后面会讲到EOD语句，它能告诉我们如何找出还有多少数据空间可用。

COMX — 35 也提供处理字符串的能力，可有26个字符串变量（从A $到Z $）每个字符串最多可包含127 个字符，另外还包含字符串数组，它使得程序员可用一些数自变量作为字符串数组的下标，因此可以用A $（N）来表示字符串，其中N 是表达式（或数），最大値是255，如果所用的数大于255，会出现错误的结果。

沒有自变量的字符串就假定它的自变量是０，

例如：A＄　就等价于A＄(０)

还有一点很重要的是，对字符串不需要任何大小规定，每产生一个字符串，存贮器就自动分配一次，还要注意，产生了字符串C＄(10)並不意味着存在C＄(１)到C＄(９)，反是C＄(10)存在，而企图读C＄(１)，其结果得到错误信息，此说明C＄(１)还未产生。

5·7 控制程序流向的指令

这些指令统称为控制语句，它们实现条件转移和无条件转移，控制语句包括END, EXIP, FOR, GOSUB, GOTO, IF, NEXT, RETURN和WAIT。

END

这个语句（程序的结尾）用作停机或结尾，它终止程序的执行並让COMX－35返回到直接执行方式，在COMX－35 BASIC程序中，END语句可放在任何位置和使用任意多次，如果它在程序中已经是最后一个语句，那么愿意的话，也可以将它删掉。

EXIT 表达式

这个语句是无条件地转移到表达式所规定行号，可用它来提早跳出FOR/NEXT循环或子程序，有一点应该十分注意的，如果子程序或FOR/NEXT循环是嵌套的，则EXIT语句所规定转移控制的行号应位于下一层嵌套之內，例如，如果FOR/NEXT循环有四层相套，而且希望提早转示第四个FOR/NEXT循环，则COMX－35 BASIC以为在EXIT 被执行之后是在第三个FOR/NEXT循环內的某个地方，然后EXIT 又可取第二个FOR/NEXT循环中的某个行号，等等，这个语句可用来代替标准的GOTO语句，並可用来清除由FOR/NEXT或子程序调用所产生的所有变更了的堆栈指示器，例子如下：

```
10    FOR I=1 TO 10
20    FOR J=1 TO 5
30    A=B+C
40    IF A=15 THEN EXIT 60
50    NEXT J
60    PR A
70    NEXT I
```

FOR变量＝表达式1 TO，表达式2 STEP，表达式3。

这个语句先赋给变量以表达式1的值，然后程序一直执行到NEXT语句，这时变量的值加上表达式3的数值，该数值可正可负。再计算（表达式2－变量）的值並与步长（STEP）的符号相比较，我们约定0算正号，如果符号相同，则重新从FOR，语句后面的第一个语句执行并继续过循环，在循环过程中，用戶可用适当的方法（例如用LET语句）来修改变量名的值，还应注意，变量名的型式就是计算表达式1，表达式2和表达式3的型式，如果省去STEP表达式3，则认为其步长是1（参3·9节）。

GOSUB表达式

这个语句（子程序调用）与GOTO语句基本相同，只是这时程序记住GOSUB出现的地方，当COMX－35 BASIC遇到RETURN语句时，就返回到紧接在GOSUB语句后面的语句去执行，用这种方法子程序可多层套用，只要存贮器允许（随着GOSUB套用次数的增加，堆栈也增加）下面是使用GOSUB语句的例子。

```
5     A=2000
10    GOSUB 1000：GOSUB 1000
20    GOSUB A
30    END
1000 PR "DONE--"
1010 RETURN
2000 PR "FINISHED"：GOSUB 3000：GOTO 1010
3000 PR "COMPLETE"：RETURN
```

本例最后打印出

```
DONE--
DONE--
FINISHED
COMPLETE
```

也可参考3·10节

GOTO 表达式

这个语句是无条件转移，马上转移到表达所指定的行号去执行，如果实际不存在此行号，则给出错误信息，例子如下：

```
10    GOTO 50
10    GOTO A+B
10    GOTO 100*（A-B）
```

IF 语句

它可取下列两种形式：

IF 对照字符串<>字符串表达式 THEN 语句

IF 表达式（关系运算符）表达式 THEN 语句

这个语句是条件语句，但在COMX-35 BASIC中它不像其他BASIC版本仅是条件转移，它首先检验条件，如果条件满足，就执行语句或一组语句（用冒号分开）如果条件不满足，那末继续执行下一行号的语句，在比较字符串关系式时，只能用符号和不等号的关系，在算术表达式情况下，下列关系运算符号都可用

	=	等于
	<>	不等于
>	>	大于
<	<	小于
<	>=	大于或等于
<	<=	小于或等于

第一个表达式的型式限定了第二个表达式的型式，下面是条件语句的例子：

```
10  IF A=B THEN PR  A:WAIT(200):CPOS(0,0):CLS:
    GOTO 200
20  PR B
```

这里，如果A的值等于B的值，那么就打印出A的值，然后停一段时间，並清除显示屏，再继续执行200行，如果A不等于B，那么打印B的值，並继续执行20行后面的语句。

附带说明一点，键词THEN可随意省略，下面是一种应防止的通病。

```
IF  A>N THEN 200
```

在THEN右边，如果有內容的语应该是可执行的语句，但数200並不是可执行的语句，正确的写法应是：

```
    IF A>B  THEN GOTO 200
或  IF A>B GOTO 200
```

下面再给出一些可行的例子，注意最后一例中的多重条件，如果任一条件不满足，就继续执行下一行号语句，如果所有条件都满足，则由GOTO语句转到执行500行语句，

```
10 IF A(2)*B(1)>=C*SIN(A) PR A:GOTO 50
10 IF A$(2);nn"LIST"  LET B=3:GOTO 100
10 IF B$(A+B)=MID$(A$,2,4)PR "OK"
10 IF MID$(A$(5),2,4)="EBCD"PR"MATCH":
   GOTO 500
10 IF A$=B$+C$ THEN GOSUB 200:CLS:PR
   "DONE"
10 IF A=INT(S/5) IF B>10 IF C<5 GOTO 500
```

NEXT

NEXT 简单变量

这个语句在FOR语句中已说过，是用来关闭FOR／NEXT循环的，如果省去变量名，则NEXT语句就返回到在它前面的FOR语句並继续执行，如果包含有变量名，则COMX－35 BASIC要检查它是否与前面的FOR语句所使用的变量名相同，如果不一致，就发生错误信息。

RETURN

这个语句，（从子程序调用返回）标记子程序结束，应返回到刚被执行的GOSUB语句后面的语句去执行。

WAIT（表达式）

这个语句提供一种在程序执行中暂停的方法，暂停时间的长短与表达式的值成正比：1相等于CPU时钟128000，表达式的值为1的暂停时间约6.4毫秒（即128000×1/(2×10)），下面是应用WAIT语句的例子：

```
10    INPUT A$(1)
20    PR "MESSAGE IS";A$(1)
30    WAIT(500):CLS:GOTO 10
```

这个程序使得在清除显示屏之前能有一段时间（500×6.4毫秒）看到该信息，並可不断执行。

指令语句

指令语句是系统指示语句，指令语句包括CLD, CLS, CPOS, EDIT, EOD, EOP, FORMAT, LIST, NEW, RENUMBER, RUN, RUN+和TRACE。

CLD

这个语句（清除数据）执行时清除所有的字符串和数组，它是通过重置字符串和数组指示器到原始状态来实现的。

CLS

清除现有光标位置以下的显示屏，如果要清除整个显示屏，重打CPOS(0,0)：CLS

CPOS（表达式，表达式）

这个语句（光标位置）是移动光标到括号内两个表达式所指定的屏上的位置，第一个是规定其行位置（0到23），第二个指定其列位置（0到39），CPOS 可以与CLS 一起使用来清除屏的一部分，与PPINT一起使用来将信息或标题从屏上的指定位置开始显示，或与SHAPE一起使用来产生彩色图形，例如：

```
10    CPOS（11,0）
20    CLS
RUN
```

将清除屏的下半部分，

```
10    CPOS（11,12）
20    PRINT "THIS IS A TEST"
RUN
```

将在显示屏的中间显示该信息。

EDIT 表达式

这个语句打开表达式所调用的行号並使用戶可以逐个字符进行修改，详情已在3·11节中给出。

EOD

EOP

EOD语句（数据结尾）打印出数据（数组和字符串）空间的结尾的十六进地址，EOP语句（程序的结尾）类似于EOD，执行时（一般在直接执行状态）自动打印出当时程序空间结尾的十六进地址，用戶可在程序进程中随时打这个命令看看已存入存贮器中的程序的结尾在那里，下面例子：

EOP：EOD

可给出

■4411
■411A

这里给出的数是十六进制的。

这个例子说明程序的结尾在4411（十六进），数据的结尾是441A（十六进），EOD仅是在执行程序产生了数据之后才是有效的，（即是执行DIM语句之后）

FORMAT N

这个语句指定数区大小（即是打印数值数据的地方大小），在FORMAT语句后面的所有PRINT语句，不管是打印变量的值或数，都按指定的数区大小，例如：

```
10   A=12345.6
20   FORMAT 7
30   PR A
40   PR 67890.1
RUN
```

将给出

```
12345.6
67890.1
```

如果行20改为

```
20   FORMAT 6
RUN
```

其结果是

```
******
******
```

如果40行改为

```
40   PR -67890.1
```

其结果是

```
******
-*****
```

总之，如果正数的数区大小大于N，将打印出N个星号，如果是负数，将打印出一个负号和N－1 个星号。

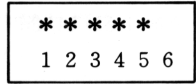
N的范围从 1 到15

FORMAT 0的意义是解释FORMAT的约束。

例如：

```
10   FORMAT 5
20   PR          123456
30   FORMAT 0
40   PR          123456
RUN
```

将给出



LIST

LIST 表达式

LIST 表达式，表达式

这个语句（列程序清单）若不需表达式，将列出用戶空间中的整个程序，如果给出一个表达式，则仅列出一行，其行号就是表达式的值，如果给出两个表达式，中间用逗号分开，则列出从第一表达式所指定的行到第二表达式所指定的行，如果表达式的值不是实际存在的行号，那末就列出在此值后面最接近的行号的那一行。（也可见3·3节）

NEW

这个语句将COMX－35 BASIC，用戶空间和数据空间全部初始化，即是清除用戶所产生的COMX－35 BASIC程序並预置所有的字符串和数组指示器。

RENUMBER

RENUMBER N

RENUMBER语句使用戶可按给定的增量重新改写程序中的行号，如果在RENUMBER语句中沒有自变量，增量就取为10，即行号为10，20，30这样往下去，若给定n，则以它为初始行号，如果n的值超过256，就都调整为256，显示出来还出现一个表示“计算转移”数的警告信息。“计算转移”是要求计算机转移或跳转到另一个其行号还未决定的行的语句，例如，考虑下面的程序

```
10    FOR A=1 TO 5
15    N(A)=0
20    NEXT
25    GOTO 10
```

```
RENUMBER  20
```

将给出信息

```
0    COMP  BR
```

同时LIST就给出

```
20    FOR A=1 TO 5
40    N=0
60    NEXT
80    GOTO 20
```

注意各行已被重新改号，另外，语句“25 GOTO 10”变成“80 GOTO 20”，用为老的行号10已改为20，“GOTO 10”还是一个“计算转移”（故有信息“0 COMP BR”），因为计算机能够决定要转移或跳转为行。

考虑另一个例子，其中用“GOTO 5 * 2”（或“GOTO B”）代替“GOTO 10”

```
10   FOR A=1 TO 5
15   N(A)= 0
20   NEXT
25   GOTO  5 * 2
```

```
RENUMBER 20
```

如给出信息

```
1    COMP BR
```

同时 LIST 将给出

```
20   FOR A=1 TO 5
40   N(A)= 0
60   NEXT
80   GOTO  5 * 2
```

有一个"计算转移"的警告信息指示我们，在这种情况下，有一行，即行80，由于改写行号的结果会出现错误，"GOTO 5 * 2"（和GOTO B）是"计算转移"的例子。

RUN

这个语句置COMX－35 BASIC以执行程序的状态COMX－35 BASIC找出最小的行号，並按行号顺序开始逐行执行，不过，在开始执行之前，RUN语句要清除所有数组和字符串数据空间以存放新数据。

RUN 表达式

这个语句是从表达式所指定的行号开始执行，它与不带表达式的RUN类似都是置COMX－35 BASIC以执行状态，如果表达式所指定的行号实际不存在，则产生一个错误代码，另外，由于这种RUN语句不清除数据空间，故用戶可执行带有先前所产生的数据（字符串或数组）的程序。

RUN+

这个语句检查用戶的程序並用“绝对地址转移”代替“解释转移”，然后开始执行，这一步大大提高了速度，经过最初一轮RUN+之后，以后RUN就以这种较快的方式执行，但如果程序经过编辑，则系统就自动转回较慢的状态。

用RUN+转变过的程序不是以这种形式存贮的，因为已指定绝对地址。

（注意：执行RUN+时，只要BASIC解释程序遇到一个“分支或转移语句”，比为“GOTO 10”它就去找存有行号“10”的绝对地址，从而将语句从所谓“解释转移”转换成“绝对地址转移”。

像“GOTO B”这样的语句不是“解释转移”，因为已经知道要转移到的行号已存贮在符号地址B的地方）。

TRACE 表达式

如果跟在后面的表达式的值非零，则“跟踪”起作用，就是说，COMX－35 BASIC 要执行的每一行都给显示屏送下列信息：

TR（行号）

屏上的数字是被执行的每一语句的行号，这样就使用戶能够“跟踪”程序的流向，这在调试阶段对保证程序能达到预定目的特别有用。

如果执行的TRACE语句后面跟着的表达式的值是0，则跟踪停止，这些语句可放在COMX－35 BASIC程序的任何地方，对TRACE语句来说非常有用的地方是在一个中断程序中。

用一个 BASIC 程序试试下面命令：

```
TRACE(1)：RUN
```

5·9 注释和定义语句

注释和定义语句使程序员可在程序中插进注释和配置存贮单元，即定义程序的开始地址，给数据分配存贮单元等一类事情，注释和定义语句包括DEFINT, DEFUS, DEG, DIM, FIXED, RAD和REM。

DEFINT

DEFINT 变量名

这个语句（定义整型变量）如果不带变量名，执行时所有变量（A－Z）和所有数组（A(表达式)──Z(表达式)）都选做浮点型，如果在DEFINT语句中包含有变量名，则从A起到此变量名（包括此变量名）的所有变量和数组都取为整形，例如

```
DEFINT  D
```

就定义以A，B，C，D为名称的变量和数组为整形由于NEW语句总是将所有变量恢复为浮点型，因此，如果在程序中要使用整形变量，就应提早在程序中出现DEFINT语句。

还要注意，如果所用的DEFINT是不带变量名的，就必须以一个回车结束，就是说，在同一行不可再用冒号来连接其它的指令。

DEFUS 表达式

这个语句（定义用戶空间的入口）是用来使程序空间的入口在存贮器中往前移动，如图5·9·1所示，它使程序员能在存贮器中创造一个“空穴”用的存贮机器语言程序，表达式定义程序空间应从何处开始，在COMX－35 BASIC中，用戶空间是在十六进地址4400处开始，表达式的值必须大于4400（十六进），COMX－35 BASIC将使表达式舍取到仅给出整页的移动量，如果企图在小于4400（十六进）的地址定义作为用戶空间，则COMX－35 BASIC将会“自我毁灭”就是说，由于程序写过系统区（见图5·9·1），BASIC解释程序不再能正常工作，每执行一次DEFUS语句，只有用另一个到4400（十六进）的DEFUS才能使程序空间返回到4400（十六进）该语句破坏了当前在存贮器中的用戶程序。

对移动用戶程序空间的入口有兴趣的一种情况是在PSAVE语句中，如果在移动过的地方已产生一个程序，同时还有有关的一些机器语言程序，PSAVE语句会录存从4400（十六进）到程序空间结尾的全部內容，包括机器语言程序和有关的COMX－35 BASIC程序，随后的P-LOAD会回输入上述的一切，包括机器语言程序，同时也重新定义用戶空间的入口到文件产生的地方，不需要管理操作。

DEFUS ■6D00 或
DEFUS ■6DCC 移动用戶空间的入口到6D00
DEFUS ■4400 移动程序空间的入口回到初始位置。

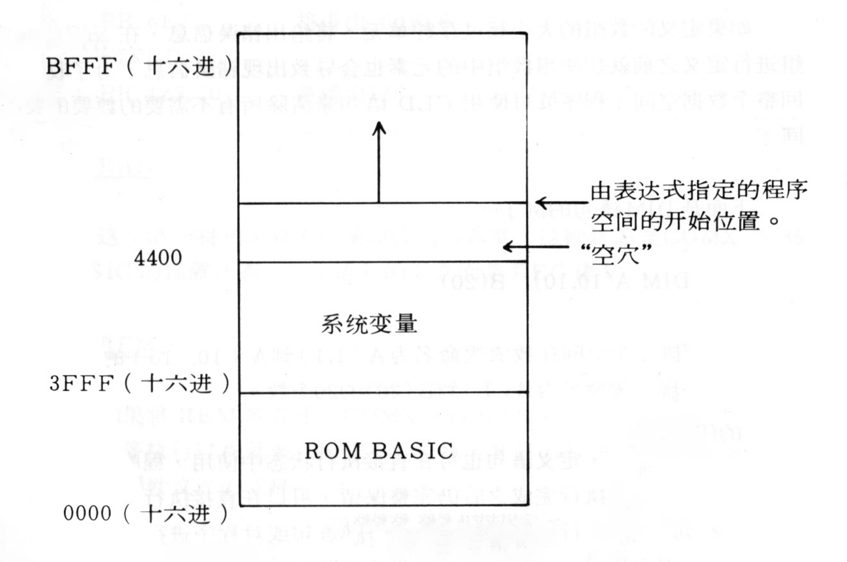


图 5·9·1 使用 DEFUS 往上移动程序空间的开始位置

DEG

这个语句使用角函数的计量单位取作角度。COMX－35 BASIC的预置状态是弧度制，这个语句的对立面是 RAD 语句。

DIM变量表

这个语句（定义数组）是用于给数値数组预定存贮单元，这些数组可以是一维或二维，表达式规定在一维或二维数组中究竟有多大，请看前面关于数组及其大小限制的讨论，数组的型式（浮点或整型）依赖于变量名是定义为整型还是浮点型，例如，如果所有变量A－Z都是浮点型，那么所有数组也将是浮点的，（与规定其大小的表达式的型式无关）如果变量A，B和C定义为整型，那么以A，B和C为名称的数组（如果使用的话），也将定义为整型，多个数组可以用同一个DIM语句来定义，只要每个数组用逗号分开。

如果定义的数组的大小超过存贮单元，将给出错误信息，在未对数组进行定义之前就想使用数组中的元素也会导致出现错误信息，为了收回整个数据空间，程序员可使用 CLD 语句来清除所有不需要的数据空间。

下面是DIM语句的例子。

DIM A(10,10)　B(20)

上面例子安排一个空间存放依次命名为A（1,1）到A（10, 10）的100个数（10×10）和命名为B（1）到B(20)的20个数。

如果需要的话，定义语句也可在直接执行状态中使用，程序所产生的任何数组在程序执行完成之后仍完整保留，可以在直接执行状态用PRINT语句来查询数组的內容在执行CLD语句或对程序进行编辑之前，数组都一直完整保留，请注意，如果执行“RUN表达式”，因为並沒有清除数据空间，故数组不会被清去，在这种情况下，如果再定义就会导致错误信息，正因为这样，程序员如果希望使用前面所产生的数据，就应从DIM语句后面的行号开始执行，数组数据和字符串数据一样，直接存在用戶空间的后面，对原来程序进行编辑就会改变用戶空间，因而就破坏了数据，应说明的是，一个数组可以不破坏其他数组或字符串而重新定义。

FIXED 表达式

这个语句执行时规定打印浮点数和整数的格式，表达式的值规定小数点右边要打印多少位，位数不足用 0 补齐，如果需要，数字可进行舍入，表达式的值应在 0 到 6 之间，如果给定的数大于 6 ，则COMX－35 BASIC 就恢复到输出数字的通常格式（即FORMAT 0)例如

```
FIXED 2
PR 123          将得出123.00
PR 123.567      将得出123.57
PR 6E7          将得出 .60E08
FIXED 7
PR 123.50       将得出123.5
```

RAD

这个语句将角函数的计量单位选为弧度，这种状态是COMX － 35 BASIC 的预置状态，这个语句的对立面是 DEG 语句。

REM

如果将 REM 放在任一COMX－35 BASIC行的开头时，这整个一行在程序执行过程就被列出来不予理睬，其目的是使程序员可插入一些作为说明性文件的注释。

5·10 程序数据语句

程序数据语句使程序员能在程序中嵌进数据表，並提供一种在程序执行过程读数据的技巧，程序数据语句包括 DATA，READ 和 RE-STORE。

DATA 数据表

DATA语句包含给READ语句使用的数据，DATA语句中的每一元素必须用逗号隔开，任一字符串都必须用引号括起来，DATA语句可放在COMX－35 BASIC程序的任何地方，下面是一些DATA语句的例子:

```
   1 DATA  2*A, A$(1),"HELLO", B
1000 DATA 1,2,3,4
5000 DATA SIN(45),SIN(60),SIN(90)
```

READ变量表

READ语句是用来从DATA语句中读出数据並将数据赋给各个变量，每当READ语句再需要数据时，内部COMX－35 BASIC指示器就移到DATA语句的下一片数据，当一个DATA语句的数据用完时，COMX－35 BASIC就往下面寻找另一个DATA语句，如果再也找不到，就显示错误信息说明数据不够用，重要的是要记住READ语句中的变量名或字符串名和DATA语句中相应的数据，它们在形式上应相匹配：字符串应对字符串，等等，下面是一个DATA／READ语句对的例：

```
 5  DIM A(4)
10  DATA 10,20,30,40
20  FOR A=1 TO 4
30  READ A(A)
40  NEXT
50  READ A$(1),B,C
60  PR A$(1);B*C
70  DATA "B*C IS EQUAL TO" 20,30
```

上述例子的结果是，存进数组A的是

A(1)=10, A(2)=20, A(3)=30, A(4)=40

行60将绘出

B*C IS EQUAL TO 600

RESTORE

这个语句将上述的COMX－35 BASIC指示器重新拨回到程序中第一个DATA语句的开头，这样就可在一个程序中多次使用DATA。

5·11 输入／输出语句

I／O（输入／输出）语句是在程序和程序用戶间提供一个联接口，I／O语句包括INPUT，PRINT和PR。

INPUT变量表

当COMX－35 BASIC遇到INPUT语句时就停下来，並在显示屏上给出一个问号“？”作为提示符，然后等待用戶回答。INPUT语句后面可以是一个变量名表，各变量名必须用逗号分开，在回答提示符“？”时，用戶可答以表达式或用逗号分开的一组表达式。如果在INPUT语句中包含三个变量名，而用戶仅回答二个表达式，COMX－35 BASIC将送出另一个提示符（？）並等待第三个表达式。如果INPUT语句只有一个变量名而用戶回答二个或更多个表达式，COMX－35 BASIC将继续执行下去，当它遇到另一个INPUT语句时，就会检出未被使用过的头一个表达式而不再给出另外的提示符。这样继续下去，直到所有未被用作输入的数据用完为止，那时才送出另一个提示符“？”被挑出来的数据就赋给相应的变量名作为输入，变量名的型式决定相对应的输入表达式的计算方式。

注意，INPUT语句不能同时包含字符串变量名和通常变量名，还有，每个包含字符串变量名的INPUT语句，并且只能包含一个变量名，由于有这个限制，程序员在回答INPUT A＄语句时，可用一个不带行号的字符串回答，回车就标志该字符串结束。下面是一些INPUT语句的例子

```
10  INPUT A, B, C, (1)
10  INPUT A(1),A(B)
10  INPUT A$
10  INPUT B$(B)
```

下面是一些错误的INPUT语句

```
10  INPUT AB（变量之间沒有定界符）
10  INPUT A$,B$（不能有多个字符串名）
```

INPUT “字符串”变量表

这个 INPUT 语句与上面的 INPUT 语句基本相同，只是它在键词 INPUT 后面紧接一个引号，它使得能在问号提示符的前面打印出一个信息，头个引号前面和尾个引号后面都不同定界符或分隔符。

PRINT 或 PR

PRINT 打印表

PR是PRINT的缩写，在注入表中时，PR就代回PRINT，这个语句的目的是要I/O程序输出前面已定义的表达式或字符串函数，各个要输出的项必须用适当的定界符逗号或分号隔开，每种定界符都有特定的功能，如果使用逗号，则下一次就打印在下一个八格的“打印区”中。如果用分号，就没有插入空位，下一次紧接在前一次的后面，在PRINT语句末尾也可用分号和逗号来制止回车，这样在程序中遇到下一个PR-INT语句时，就从上一个语句离开的地方开始。如果 PRINT 语句只是单个 PRINT，则输出的是回车或空行，有两个只能在 PRINT 语句中使用的函数：TAB（表达式）和CHR $（表达式），关于这两个函数已分别在5·13节和5·5 节做了说明，下面是一些PR语句的例子：

```
PRINT 5
PRINT A
PR (A+B)/C
PR A,B,1234,C(5)
PR "THE VALUE OF A=";A
PR A$(1)+A$(2)
PR A$(2),A,B;
PRINT MID$(A$,1,2)
PRINT TAB(A+B);10;TAB(30);E
PR A,
PR
```

注意，当 PRINT 语句中出现多于一个算术表达式时，每个表达式的计算可取不同方式。

5·12 机器语言子程序语句

机器语言子程序语句使程序员可在其程序中包含机器语言代码，以利用处理机的特殊性能或提高实时应用的速度，机器语言子程序语句是CALL。

CALL（表达式1）

CALL（表达式1，表达式2）

CALL（表达式1，表达式2，表达式3）

这个语句提供COMX－35 BASIC与机器语言程序设计之间的链接，它用于机器语言子程序调用，它使执行转移到由表达式规定其地址的机器语言子程序，编写机器语言程序应该记住下面的规则：

(1)进入子程序的程序计数器是R3
(2)用D5（SEP R5）指令返回COMX－35 BASIC
(3)机器语言程序可随意使用R8, RA, RC, RD和RE。如果要使用其他寄存器，首先应存到堆栈并在返回COMX－35 BASIC之前复原。

注意，不可使用标准的调用和返回方法（SCRT），因为它会破坏寄存器F（RF.1）的上部。

(4)COMX－35 BASIC已建立了调用和返回的规约，不需要再用机器语言子程序来重复。

(5)可以使用堆栈（由R 2指示），只要一离开就返回就可以。

每个表达式（表达式1，表达式2，表达式3）都可用整型式浮点型表示。COMX－35 BASIC会自动将其转换成整型，如果有表达式2，则将其值变成R 8中的机器数，第二部分数据也可送到寄存器R A中的机器语言子程序，其值就是表达式3的值，对应用程序要给R D预置定时常数。

5·13 I/O 用途

这组函数包括MEM，PEEK 和 TAB

MEM

这个函数使用戶能确定还剩多少个存贮单元，执行时，MEM给出一个代表在数据（字符串和数组）的结尾和通常堆栈的结尾之间留下的存贮字节的个数的十进数，考虑到程序执行过程堆栈的增长，实际送回的数量用256 缩减。例如，要在屏上得到其值用戶就应打。

```
PR  MEM
```

PEEK（表达式）

这个函数送回与由该表达式所确定的地址那个存贮单元的內容等价的十进数，例如

```
10  A=PEEK(16842)
20  IF A=8 THEN PRINT "NTSC MACHINE"
30  IF A=9 THEN PRINT " PAL MACHINE"
```

现在变量A 包含存贮在存贮单元16842 的“內容”。这是二个“系统变量”，它告诉我们你的COMX－35是PAL机，还是MTSC机。

TAB（表达式）

这个函数与上面两个函数不同，不送回任何数值，它只能在 PRINT 语句中才可使用，它将光标移到由该表达式的值确定的水平位置上，光标的新位置是相等于第 0 列而定的，而不是相对于先前的位置而定的。然后就从所指的地方继续打印。

下面是一些例子：

```
PR  TAB(10); 1
```

将在第10列打印出1

PR TAB(10)；1；TAB(20)；2

将在第10列打印出1和在第20列打印出2

PR TAB(A)；"*"

将在第A列打印出一个星号"*"

5·14 机器语言用途

这一类只有唯一的函数USR

USR（表达式1）
USR（表达式1，表达式2）
USR（表达式1，表达式2，表达式3）

这个函数的作用很像上一段所说CALL语句，不同的是USR是一个函数，它被用作表达式的一部分，当遇到USR时，就往储存在表达式1所指定位置的机器语言程序调用子程序，数据可以与CALL完全相同的方法送到子程序。

请注意：USR和CALL是使用1802微处理器规约和寄存器，关于1802 CPU的详细解释请参考1802程序设计手册。

第 六 章

程序设计例子

第六章　程序设计例子

这一章收集一些程序，除了这些程序本身有用外，也作为例子说明某些BASIC指令的应用对每个程序，除了提供其BASIC源程序表和打印运行的屏幕显示外，对其中某些程序设计的技巧也做了注解。

6·1 关于CAI（计算机支持指令）的第一个例子一

用实例说明COMX－35的COLOR，SCREEN和MUSIC 指令的用法。

6·1·1 程序表

```
10   REM THIS PROGRAM IS A TUTORIAL TO
20   REM SHOW HOW TO COMMAND COMX 35
30   REM TO CHANGE THE COLOR OF THE
40   REM CHARACTERS, AND OF THE SCREEN
50   REM AND HOW TO PRODUCE SOUND
60   REM EFFECT
70   REM
75   CPOS (0,0):CLS
80   SCREEN (3): COLOR (8): CPOS (1,7): PRINT "HI!
     I AM THE CLEVER COMX 35!"
90   CPOS (3,4): PRINT "I LIKE TO SHOW YOU"
100  CPOS (5,4): PRINT "HOW TO DO THE FOLLOWING:"
110  CPOS (7,4): PRINT "     1. HOW TO CHANGE THE
     COLORS"
120  PRINT "     OF THE CHARACTERS DISPLAYED"
140  CPOS (10,4): PRINT "     2. HOW TO CHANGE THE
     COLOR"
150  PRINT "    OF THE SCREEN."
160  CPOS (13,4): PRINT: "    3. HOW TO PRODUCE SOUND
     EFFECTS."
170  CPOS (15,2): INPUT "WHICH OF THE ABOVE WOULD
     YOU LIKE TO KNOW? 1,2 OR 3. ANSWER 0 TO END"
     K
175  PRINT
177  IF K=0 THEN GOTO 710
```

```
180  GOTO K * 200
200  SCREEN (1): COLOR (7): CPOS (0,0): CLS
204  CPOS (3,8): PRINT "TO CHANGE THE COLOR OF THE"
205  CPOS (4,11): PRINT "CHARACTERS, WE USE THE"
210  CPOS (5,11): PRINT "COMMAND COLOR(N) WHERE N"
215  CPOS (6,11): PRINT "DETERMINES"
220  CPOS (8,8): PRINT "  THE COLOR OF THE COMPUTER"
225  CPOS (9,9): PRINT "  OUTPUT CHARACTERS, AND"
230  CPOS(11,8): PRINT "  THE COLOR OF THE KEYBOARD"
235  CPOS (12,9):PRINT "  INPUT CHARACTERS"
240  CPOS (14,8): PRINT "PLEASE TYPE"
245  CPOS (16,8): PRINT "  COLOR (N)"
247  CPOS (18,8): PRINT "(WHERE N CAN BE 2,3,4,5,6,7,8)"
250  CPOS  (20,8): PRINT  "AFTER THE   ? SIGN. THEN.
     PRESS"
255  CPOS  (21,11):  PRINT  "THE  CR  KEY"
260  INPUT  A$
265  B$=MID$(A$,7,1)
270  C=FVAL  (B$)
275  COLOR  (C)
280  PRINT  "TYPE  ANY  KEY,  FOLLOWED  BY"
283  PRINT  "PRESSING  CR  TO  REVERT  TO"
285  PRINT  "ORIGINAL  COLOR"
290  INPUT  A$
295  COLOR (12)
300  PRINT  "N  CAN  HAVE  VALUE  FROM  1  TO  12"
305  PRINT  "INCLUSIVE.  YOUR  MANUAL  WILL"
310  PRINT  "TELL  YOU  WHAT  COLOR"
315  PRINT  "COMBINATION  FOR  EACH  VALUE"
320  PRINT  "OF  N."
325  CPOS  (0,0):  CLS:  GOTO  80
400  SCREEN (7): COLOR (12): CPOS (0,0): CLS
404  CPOS  (6,4):  PRINT  "TO CHANGE THE BACKGROUND
     COLOR"
405  CPOS (8,4): PRINT "OF THE SCREEN, WE USE THE"
410  CPOS (10,4): PRINT "SCREEN(M) COMMAND."
415  CPOS (13,4): PRINT "FOR EXAMPLE"PLEASE TYPE
420  CPOS (16,4): PRINT "  SCREEN(M)"
422  CPOS  (19,4):  PRINT  "(WHERE  M  IS  ANY  INTEGER  1
     TO 8)"
425  CPOS (21,4): PRINT "AFTER THE  ? SIGN. AND PRESS
     CR"
430  INPUT A$
```

```
435  B$=MID$(A$,8,1)
440  C=FVAL (B$)
445  SCREEN (C)
450  PRINT "PRESS ANY KEY FOLLOWED BY CR"
455  PRINT "REVERT TO THE ORIGINAL SCREEN"
460  PRINT "COLOR"
465  INPUT A$
470  SCREEN (1)
475  PRINT "M CAN HAVE VALUES OF 1 TO 8"
480  PRINT "INCLUSIVE. YOUR MANUAL WILL"
485  PRINT "TELL YOU THE SCREEN COLOR"
490  PRINT "CORRESPONDING TO EACH"
495  PRINT "VALUE OF N"
500  CPOS (0,0): CLS: GOTO 80
600  SCREEN (2): COLOR (5): CPOS (0,0): CLS
604  CPOS (3,4): PRINT "MUSICAL NOTE IS GENERATED
     USING"
605  CPOS (5,4): PRINT "THE MUSIC (X,Y,Z) COMMAND"
610  CPOS (8,4): PRINT "  X=1 TO 7 FOR DO TO TEE"
615  CPOS (10,4): PRINT "  Y=1 TO 8 FOR OCTAVE"
620  CPOS (12,4): PRINT "  Z=0 TO 9 FOR AMPLITUDE"
625  CPOS (15,4): PRINT "TRY, FOR EXAMPLE, TYPING,"
630  CPOS (18,4): PRINT "  MSUIC (X,Y,Z)"
635  CPOS (21,4): PRINT "AFTER THE  ? SIGN. THEN PRESS
     CR"
640  INPUT A$
645  X$=MID$ (A$,7,1)
650  Y$=MID$(A$,9,1)
655  Z$=MID$(A$,11,1)
660  X=FVAL (X$)
665  Y=FVAL (Y$)
670  Z=FVAL (Z$)
675  MUSIC (X,Y,Z)
680  PRINT "HIT ANY KEY FOLLOWED BY CR TO"
685  PRINT "STOP THE MUSIC NOTE"
690  INPUT A$
700  MUSIC (0,0,0)
705  CPOS (0,0): CLS: GOTO 80
710  SCREEN (1): COLOR (12): END
```

6·1·2 程序设计要点

(1) 10行到60行中使用REM是用来说明程序的目的。

(2) 75行是清除显示屏以使以后可显示。

(3) 80行，注意其中使用SCREEN和COLOR来改变显示的颜色

(4) 80行到160行，注意其中使用CPOS来给要显示的信息定位置。

(5) 170 行，这个INPUT 语句输出一个信息向用戶解释如何与计算机“对话”，这里要求用戶在答案1·2·3或0中任选一个。

(6) 260 行，计算机从用戶接受形式为COLOR(N)的字符串A$，假定用戶输入的是COLOR(6)。

(7) 265 行到275 行，命令计算机执行用戶要求的COLOR(6)指令，这三行刚好使用MID$和FVAL函数，265 行是从字符串COLOR(6)中抽取字符6，因此，B$就等于字符6的ASCII代码，在270 行，FVAL计算字符6的值並将其值送给C，这时C = 6，由COLOR(C)所改变的颜色与要求的一致，在435 行到445 行，和645行到675 行都重复同样的方法。

6·2 关于CAI(计算机支持指令)的第二个程序

例子——用COMX－35计算数学表达式的值

6·2·1 程序表

程序表

```
10    REM THIS PROGRAM IS A TUTORIAL TO
20    REM SHOW HOW COMX MAY BE USED
30    REM AS A CALCULATOR TO EVALUATE
40    REM COMPLEX MATHEMATICAL EXPRESSIONS
60    CPOS (0,0): CLS
```

```
70   CPOS (4,5): PRINT "I WISH TO SHOW YOU MY MATH"
80   CPOS (5,6): PRINT "   ABILITY"
90   CPOS (7,5): PRINT "LET'S BEGIN WITH ADDITION."
100  CPOS (8,5): PRINT "THE OPERATOR IS +."
130  CPOS (10,5): PRINT "PLEASE TYPE"
140  CPOS (12,5): PRINT "3+4"
150  CPOS (14,5): PRINT "AFTER THE? SIGN."
160  CPOS (17,5): PRINT "THEN, PRESS THE CR KEY"
165  CPOS (19,5): PRINT "WATCH OUT FOR THE ANSWER
     IN"
167  CPOS (20,6): PRINT "  THE NEXT LINE"
170  INPUT A$
180  PRINT FVAL (A$)
182  INPUT "HIT THE RETURN KEY AND PRESS CR TO
     CONTINUE" K$
184  CPOS (0,0): CLS
190  CPOS (8,4): PRINT "NOW, LET'S TRY SOMETHING
     MORE"
200  CPOS (9,6): PRINT "COMPLICATED"
210  CPOS (11,4): PRINT "TYPE AFTER THE? SIGN"
220  CPOS (13,4): PRINT "  12+14+61"
221  CPOS (15,4): PRINT "THEN, PRESS CR FOR THE
     ANSWER"
224  PRINT
226  PRINT
230  INPUT A$
240  PRINT FVAL (A$)
242  INPUT "HIT ANY KEY AND PRESS CR TO CONTINUE"
     A$
250  CPOS (0,0): CLS
260  CPOS (8,4): PRINT "THE OPERATOR IS–."
270  CPOS (11,4): PRINT "PLEASE TYPE"
280  CPOS (13,4): PRINT "  9–5  "
290  CPOS (15,4): PRINT "AFTER THE? SIGN. PRESS CR"
300  PRINT A$
310  PRINT FVAL (A$)
315  INPUT "HIT ANY KEY AND PRESS CR TO CONTINUE"
     K$
320  CPOS (0,0): CLS: CPOS (7,8): PRINT "NOW, TRY THE
     FOLLOWING"
330  CPOS (9,8): PRINT "TYPE AFTER THE? SIGN"
340  CPOS (12,8): PRINT "  5+6–7  "
345  CPOS (15,8): PRINT "AND PRESS CR FOR THE
     ANSWER"
```

```
350  INPUT A$
360  PRINT FVAL (A$)
365  INPUT "HIT ANY KEY AND PRESS CR TO CONTINUE"
     K$
370  CPOS (0,0): CLS: CPOS (5,5): PRINT "NOW, LET'S
     TRY MULTIPLICATION"
380  CPOS (7,5): PRINT "THE OPERATOR IS*"
390  CPOS (9,5): PRINT "PLEASE TYPE"
400  CPOS (12,5): PRINT "9*7"
410  CPOS (15,5): PRINT "AFTER THE? SIGN. PRESS CR"
420  INPUT A$
430  PRINT FVAL (A$)
435  INPUT "HIT ANY KEY AND PRESS CR TO CONTINUE"
     K$
440  CPOS (0,0): CLS: CPOS (5,5): PRINT "NOW LET'S TRY
     DIVISION"
450  CPOS (7,5): PRINT "THE OPERATOR IS /"
460  CPOS (9,5): PRINT "PLEASE TYPE"
465  CPOS (12,5): PRINT " 63/7 "
470  CPOS (15,5): PRINT: AFTER THE? SIGN. PRESS CR"
480  CPOS (17,5): PRINT "(63/7 MEANS 63 DIVIDED BY 7)"
490  INPUT A$
500  PRINT FVAL (A$)
520  CPOS (20,0): INPUT "HIT ANY KEY AND PRESS CR TO
     CONTINUE" K$
522  CPOS (0,0): CLS: CPOS(2,5): PRINT "NOW, LET'S TRY
     A MORE COMPLEX"
530  CPOS (4,5): PRINT "EXPRESSION. PLEASE TYPE"
540  CPOS (7,5): PRINT " (5+6)*(9–7) "
550  CPOS (10,5): PRINT "AFTER THE? SIGN. PRESS CR"
560  INPUT A$
570  PRINT FVAL (A$)
575  INPUT "HIT ANY KEY AND PRESS CR TO CONTINUE"
     K$
580  CPOS (0,0): CLS: CPOS(8,4): PRINT "NOW, TRY
     ANOTHER. PLEASE TYPE"
590  CPOS (11,4): PRINT " (7+4)*(19–14)/(8–3) "
600  CPOS (13,4): PRINT "AFTER THE? SIGN. PRESS CR"
610  INPUT A$
620  PRINT FVAL (A$)
625  INPUT "HIT ANY KEY AND PRESS CR TO CONTINUE"
     K$
630  CPOS (0,0): CLS: CPOS (5,5): PRINT "PLEASE FEEL
     FREE TO TRY ANY OF"
```

```
640 CPOS (7,5): PRINT "YOUR OWN EXPRESSION "
650 CPOS (10,5): PRINT "JUST TYPE IT AFTER THE? SIGN"
660 CPOS (12,5): PRINT "AND PRESS THE CR KEY"
670 INPUT A$
680 PRINT FVAL (A$)
690 INPUT "ANOTHER EXPRESSION, ANSWER Y/N"Q$
692 IF Q$="Y" THEN GOTO 630
694 IF Q$="N" THEN GOTO 720
720 CPOS (0,0): CLS: CPOS (5,5): PRINT "AFTER YOU
    FINISH THIS TUTORIAL"
730 COPS (7,5): PRINT "YOU CAN USE COMX TO COMPUTE".
740 CPOS (9,5): PRINT "ANY EXPRESSION OF YOUR OWN."
750 CPOS (12,5): PRINT "JUST TYPE PR OR PRINT.
    FOLLOWED"
760 CPOS (14,5): PRINT "BY THE EXPRESSION. THEN
    PRESS"
770 CPOS (16,5): PRINT "THE CR KEY. FOR EXAMPLE,"
780 CPOS (19,5): PRINT "PR6*5/9"
785 CPOS (23,1): INPUT "HIT ANY KEY AND PRESS CR
    TO CONTINUE" K$
790 CPOS (0,0): CLS: CPOS (5,5): PRINT "COMX CAN IN
    FACT HANDLE"
800 CPOS (7,5): PRINT "TRIGONOMETRIC, LOG, EXPONEN-
    TIAL"
810 CPOS (9,5): PRINT "AND MANY OTHER FUNCTIONS."
820 CPOS (12,5): PRINT "PLEASE CONSULT THE MANUAL"
830 CPOS (15,7): PR INT "USING THE COMX 35"
840 CPOS (20,7): PRINT "GOOD BYE! HAVE FUN"
850 END
```

6·2·2 程序设计要点

在140行中，要求用戶输入算术表达式3＋4。因此，字符串A$是3＋4，在180行中的语句PRINT FVAL（A$）求出算术表达式3＋4的值。

用这种方法可以求出更复杂的算术表达式的值，在这整个程序中，使用同样方法的还有230行，240行和300行，310行等等。

690行是一个INPUT语句，使用戶可在答案Y（对应YES）或N（对应NO）中作选择，在692行和694行中取不同的动作。



附 录

附　　录

A·1 COMX－35错误信息

00　由用戶引起的程序中断
01　在ASC或LEN函数中有语法错误
02　数组超出范围或无定义
03　维数错误
04　DEFINT有违反规定的结尾
05　在自变量中缺少括号
06　自变量超出范围
07　遇到混合型式的计算
08　以0为除数的错误，或负数的对数
09　遇到不可执行的函数
10　EXIT指令必须与FOR/NEXT或GOSUB/RETURN一起使用
11　FOR/NEXT堆栈溢出，或FOR/NEXT直接执行
12　FOR中有语法错误
13　GOSUB堆栈溢出
14　在十六进制数中有不可接受的字符
15　浮点数太大以致转换成整数或整数乘溢出
16　在条件语句中有不可接受的算符
17　INPUT或FVAL不能直接执行
18　在READ中必须有变量名或字符串名
19　在READ或INPUT中有语法错误
20　在LEN函数中有语法错误
21　在赋值语句中有语法错误
22　缺少引号
23　在LIST中有语法错误
24　在库中找不到这样的字
25　在MID$函数中有语法错误
26　在NEXT语句中发现有不可接受的变量名
27　应该不是数就是字
28　缺少运算括号
29　在POKE语句中自变量的个数错误
30　在PRINT语句中最后的字符不可接受
31　在DATA语句中有语法错误
32　数据不夠
33　在INPUT语句中找不到这样的字符串
34　在赋值语句中缺少等号

35 在字符串数组中缺乏括号
36 在USR或CALL中自变量太多
37 在CHR$函数中有语法错误
38 在二进制数中有不可接受的字符
39 行缓冲器溢出
40 输入时文件未打开
41 输出时文件未打开
42 行的结尾不可接受或语句不能执行
43 堆栈溢出
44 数的位数太多
45 在数中有不可接受的字符
46 找不到这样的行号
47 在IF语句中有不可接受的运算
48 存贮器溢出
49 在MOD语句中自变量的个数错误
50 程序太长存贮器不够存贮
51 自变量超限
52 自变量的个数错误
53 自变量的个数错误
54 字符串变量未定义
55 磁带读错
56 磁带写错
57 文件不是BASIC程序
58 文件不是BASIC数据
59 被保留的
60 被保留的
61 被保留的
62 未送ROM或ROM卡片
63 沒有足夠的存贮单元供重新编号
64 重编号设置的外号有错
65 沒有定义这样的下标变量
66 字符串超出127个字符
67 在COLOR中自变量的个数必须是1
68 在SCREEN中自变量的个数必须是1
69 在CTONE中的自变量的个数必须是1
70 在VOLUME中自变量的个数必须是1
71 在NOISE中自变量的个数必须是2
72 在TONE中自变量的个数必须是3

A·2 键符的ASCII代码和内部字符

Decimal	Hex	Key pressed	GRAPHIC Output Shape	GRAPHIC Output Color
0	0			
1	1	CNTL—A		C
2	2	CNTL—B		C
3	3			
4	4	CNTL—D		C
5	5	CNTL—E		C
6	6	CNTL—F		C
7	7	CNTL—G		C
8	8	CNTL—H		C
9	9			
10	A	LINE FEED		
11	B			
12	C	CNTL—L		C
13	D	CARRIAGE RETURN		
14	E	CNTL—N		C
15	F	CNTL—O		C
16	10	CNTL—P		C
17	11	CNTL—Q		C
18	12			
19	13			
20	14	CNTL—T		C
21	15	CNTL—U		C
22	16	CNTL—V		C
23	17	CNTL—W		C
24	18	CNTL—X		C
25	19	CNTL—Y		C
26	1A	CNTL—Z		C
27	1B			
28	1C			
29	1D			
30	1E			
31	1F			

Decimal	Hex	Key pressed	STANDARD Output Shape	STANDARD Output Color
32	20	Space Bar		
33	21	!	!	C
34	22	"	"	C
35	23	#	#	C
36	24	$	$	C
37	25	%	%	C
38	26	&	&	C
39	27	'	'	C
40	28	(	(	C
41	29	)	)	C
42	2A	*	*	C
43	2B	+	+	C
44	2C	,	,	C
45	2D	—	—	C
46	2E	.	.	C
47	2F	/	/	C
48	30	ϕ	ϕ	C
49	31	1	1	C
50	32	2	2	C
51	33	3	3	C
52	34	4	4	C
53	35	5	5	C
54	36	6	6	C
55	37	7	7	C
56	38	8	8	C
57	39	9	9	C
58	3A	:	:	C
59	3B	;	;	C
60	3C	<	<	C
61	3D	=	=	C
62	3E	>	>	C
63	3F	?	?	C

Note: CNTL—A stands for CONTROL A.

Decimal	Hex	Key pressed	Shape	Output Color
64	40	■	■	C
65	41	A	A	C
66	42	B	B	C
67	43	C	C	C
68	44	D	D	C
69	45	E	E	C
70	46	F	F	C
71	47	G	G	C
72	48	H	H	C
73	49	I	I	C
74	4A	J	J	C
75	4B	K	K	C
76	4C	L	L	C
77	4D	M	M	C
78	4E	N	N	C
79	4F	O	O	C
80	50	P	P	C
81	51	Q	Q	C
82	52	R	R	C
83	53	S	S	C
84	54	T	T	C
85	55	U	U	C
86	56	V	V	C
87	57	W	W	C
88	58	X	X	C
89	59	Y	Y	C
90	5A	Z	Z	C
91	5B			C
92	5C			C
93	5D			C
94	5E	^	↑	C
95	5F			C
96	60		ϕ	G
97	61	SHIFT–A	1	G
98	62	SHIFT–B	2	G
99	63	SHIFT–C	3	G
100	64	SHIFT–D	4	G
101	65	SHIFT–E	5	G
102	66	SHIFT–F	6	G
103	67	SHIFT–G	7	G
104	68	SHIFT–H	8	G
105	69	SHIFT–I	9	G
106	6A	SHIFT–J	ϕ	G
107	6B	SHIFT–K		G
108	6C	SHIFT–L		B
109	6D	SHIFT–M		Blk
110	6E	SHIFT–N		C
111	6F	SHIFT–O		C
112	70	SHIFT–P		C
113	71	SHIFT–Q		C
114	72	SHIFT–R		C
115	73	SHIFT–S		C
116	74	SHIFT–T	C	C
117	75	SHIFT–U	O	C
118	76	SHIFT–V	M	C
119	77	SHIFT–W	X	C
120	78	SHIFT–X		G
121	79	SHIFT–Y	C	C
122	7A	SHIFT–Z	O	G
123	7B		M	B
124	7C			
125	7D			G
126	7E		©	G
127	7F		◆	B

SPECIAL CHARACTERS (HEX 80 – HEX 8F)

Decimal	Hex	Used as output	Read from keyboard
128	80	Cursor Up (Vertical tab)	
129	81	Cursor right	
130	82	Cursor down (Line feed)	
131	83	Cursor left	
132	84	Carriage return	
133	85	Clear to end of screen	
134	86		Delete Key
135	87		Control–S
136	88		Joystick up
137	89		Joystick right
138	8A		Joystick down
139	8B		Joystick left
140	8C		Control–R
141	8D		Control–C
142	8E		
143	8F		

GRAPHIC

Decimal	Hex	Key pressed	Shape	Output Color
144	90			W
145	91			W
146	92			
147	93			
148	94			W
149	95			W
150	96			W
151	97			W
152	98			W
153	99			W
154	9A			W
155	9B			
156	9C			
157	9D			
158	9E			
159	9F			
160	Aϕ			
161	A1		!	W
162	A2		"	W
163	A3		#	W
164	A4		$	W
165	A5		%	W
166	A6		&	W
167	A7		'	W

STANDARD

Decimal	Hex	Key pressed	Shape	Output Color
168	A8		(	W
169	A9		)	W
170	AA		*	W
171	AB		+	W
172	AC		,	W
173	AD		–	W
174	AE		.	W
175	AF		/	W
176	Bϕ		ϕ	W
177	B1		1	W
178	B2		2	W
179	B3		3	W
180	B4		4	W
181	B5		5	W
182	B6		6	W
183	B7		7	W
184	B8		8	W
185	B9		9	W
186	BA		:	W
187	BB		;	W
188	BC		<	W
189	BD		=	W
190	BE		>	W
191	BF		?	W

Decimal	Hex	Key pressed	Shape	Color
192	Cϕ			W
193	C1		A	W
194	C2		B	W
195	C3		C	W
196	C4		D	W
197	C5		E	W
198	C6		F	W
199	C7		G	W
200	C8		H	W
201	C9		I	W
202	CA		J	W
203	CB		K	W
204	CC		L	W
205	CD		M	W
206	CE		N	W
207	CF		O	W
208	Dϕ		P	W
209	D1		Q	W
210	D2		R	W
211	D3		S	W
212	D4		T	W
213	D5		U	W
214	D6		V	W
215	D7		W	W
216	D8		X	W
217	D9		Y	W
218	DA		Z	W
219	DB			W
220	DC			W
221	DD			W
222	DE			W
223	DF			W

Decimal	Hex	Key pressed	Output Shape	Output Color
224	Eϕ		ϕ	Y
225	E1		1	Y
226	E2		2	Y
227	E3		3	Y
228	E4		4	Y
229	E5		5	Y
230	E6		6	Y
231	E7		7	Y
232	E8		8	Y
233	E9		9	Y
234	EA		ϕ	Y
235	EB			Y
236	EC			M
237	ED			R
238	EE			W
239	EF			W
240	Fϕ			W
241	F1			W
242	F2			W
243	F3			W
244	F4		C	W
245	F5		O	W
246	F6		M	W
247	F7		X	W
248	F8			Y
249	F9		C	W
250	FA		O	Y
251	FB		M	M
252	FC		X	R
253	FD			Y
254	FE		©	W
255	FF			M

特殊键词指令

Control – C	输入行作废
Control – I	光标向上
Control – J	光标向左
Control – K	光标向右
Control – M	光标向下
Control – R	重复最后行(到"RETURN"键)
Control – S	停止编辑

颜色略语

Blk	— 黑色
B	— 蓝色
G	— 绿色
C	— 青色
R	— 红色
Y	— 黄色
M	— 洋红
W	— 白色

A·3 磁带录音的要领

1. 使用高质量的磁带
2. 使用尽可能短的磁带
3. 使用有特别保护的电源电缆
4. 保持磁头和压带轮的清洁
5. 由于磁带的互换性应保持磁头的排列
6. 不要录在太接近带头的地方
7. 检查盒带在录音机上是否完全放好
8. 如果试过一盒磁带不好用，可试另一盒，可能是磁带上有地方坏或带盒弯曲。
9. 对装入程序要选择适当的音量
10. 骯脏的录音机会使磁带出问题
11. 仔细检查录音连接插头的接触是否良好
12. 磁带从录音机退出之前要重绕
13. 磁带应保存在原来的防尘盒中
14. 磁带应避开高温或磁场
15. 录音之前，应将磁带绕到一端
16. 有些卡式录音机常会产生问题（可能不适于某些盒式磁带或其他问题）如果某台机老是出问题，应设法换一台
17. 仔细检查 MIC 插头在录音之前是否连接好.
18. 某些磁带录音机可能要用EAR插头来放音，请查看录音机手册
19. 最好一直使用适合录音机的交流适配器
20. 如果有音调控制的话，请调到最高位置
21. 可能的话应定期对磁头和压带轮消磁，磁头磁化了会慢慢揩去磁带上的內容。
22. 拔出耳机软线听了程序的声音，如果信息头的音调失常，说明盒带或磁头有问题
23. 为了得到最好效果，应定期对你的磁带录音机消磁

A·4 COMX—35的语句和函数的索引

A·5 有关硬件的注释

A·5·1 起动自诊断序列

在初次起动或系统重置（同时按空棒和R T键）以后，COMX－35计算机会产生几声“BEEP”的声音並在电视屏幕上显示测试图样，这是自测试序列，它包含 COMX－35 计算机的很多信息，服务人员能据此判断需要服务的问题的根源。

这个测试图样也可用来调整电视接收机，它大约每隔 7 秒种改变颜色一次，这样就可调整电视机上的调谐亮度，对比度，色彩（单色和混合色）控制以得到最满意的彩色图象，在得到满意的电视图象之后，可按除空棒外的任意键，COMX－35 就显示它的软件版本的号码並准备接受你的指令。

A·5·2 PAL和NTSC制式

COMX－35 计算机特为PAL和NTSC电视系统设座两种制式，这两种制式之间有一些不同的地方，请用戶注意下面几点：

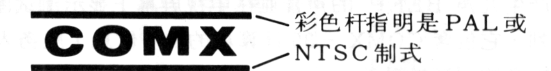
1. 在PAL制式中每个字符由 9 条水平线构成，而在NTSC中是 8 条水平线，由于在COMX －35 的NTSC制式中最后一条（第 9 条）水平线没有显示出来，故使一些图形字符串上去就不同。

在NTSC系统中，这也会影响到SHAPE指令，因为定义符号或字符的形状不必用第17和第18个数字。

2. 内部计时器在PAL制式中是按每秒50次的比率，而在NTSC制式中是每秒60次，故此，如果需要精确的时间，在 TIME（ X ）指令中就应使用不同的值。

A·5·3 PAL和NTSC制式的不同

用戶只需在开机式系统重量之后为一下测试图样就能说出COMX－35是PAL制式还是NTSC制式的计算机，两条水平线在PAL或NTSC制式中会显示出不同的颜色序列。



彩色杆指明是PAL或NTSC制式

	PAL制式					NTSC制式				
	C	O	M	X	杆	C	O	M	X	杆
图1	C	Y	M	R	G	C	Y	M	R	Y
图2	Y	C	M	B	G	Y	C	M	B	C
图3	M	C	Y	G	B	M	C	Y	G	C

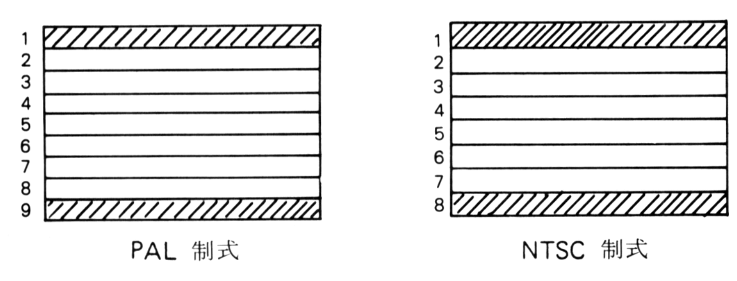
C=青色　Y=黄色　M=洋红
R=红色　G=绿色　B=蓝色

如果一个程序要都能在PAL和NTSC机上运行且得出同样结果，该程序就必须首先查悉存放在地址41CA（十六进）中的系统参数来校对机器的制式，如果其参数是9，那是PAL机，如果是8，那是NTSC机。

用戶可用PEEK指令取出这个参数，並可如下例采取适当的动作。

```
10   V = PEEK (■ 41CA) 注：找出制式並放进变量V
20   IF V = 8 THEN TIME (120) 注：在NTSC系统中延迟2秒。
30   IF V = 9 THEN TIME (100) 注：在PAL系统中延迟2秒。
40   IF V = 8 THEN SHAPE (20,'FF000000000000FF')
50   IF V = 9 THEN SHAPE (20,'FF00000000000000FF')
```

这个程序例子首先找出制式並将其放进变量V，在20行和30行中，对PAL和NTSC机要用不同的时间延迟值才能保证在这两种情况下都有精确的2秒延迟时间，在40行和50行中，程序将产生带有顶，下两根线的图符，在电视屏幕上看起来是相同的，但实际上它们分别由8条或9条水平线构成。



PAL 制式　　　　NTSC 制式

A·5·4 COMX－35计算机的调整

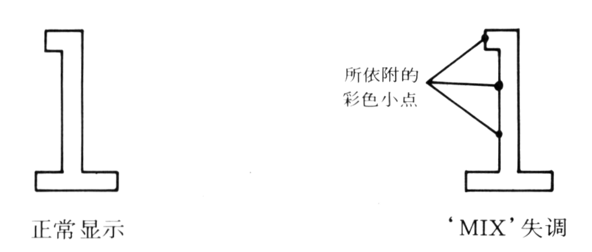
每台COMX－35计算机都已由制造厂调整到有最好的结果，如果原来的调整已被搅乱，可用小螺丝批（修理钟表用的）通过计算机底边的三个小孔做再调整。

这三个小孔分别标有MIX，SYNC和VIDO，其功能和调整步骤如下：

(1) MIX

这是用于调整彩色分载信号的频率使得亮度信号与色度信号之间的干扰最小。

如果这个控制被调乱，那么在所显示的字符的边缘会出现一些移动的或固定的彩色小点。



正常显示　　　　'MIX'失调

调整"MIX"控制的步骤

（a）打进下面的程序并使显示很多竖线。

（b）用小螺丝批小心插进标有"MIX"的小孔。

（c）观察数字"1"的边缘和彩色竖线之间的地方，缓慢地仔细转动螺丝批。

（d）调整到依附在上面的彩色点不出现为止。

注意：有些机器的电视显示会突然变成黑白，这时只要反方向转动螺丝批，彩色会重新出现。

```
NEW
10     SHAPE (#40, "FFFFFFFFFFFFFFFFFF")
20     PRINT "11111111111111111111111111111111
       1111111111111111111";
30     FOR L1=1 TO 22
40     PRINT CHR$ (107,108,64,237,235,236,192);
50     FOR I=1 TO 8
60     READ A
70     FOR D=1 TO 4:PRINT CHR$(A); : NEXT
80     NEXT
90     PRINT "        ";
100    RESTORE
110    NEXT
120    REM
130    GOTO 120
140    DATA 32,107,108,64,237,235,236,192:END

READY
:RUN
```

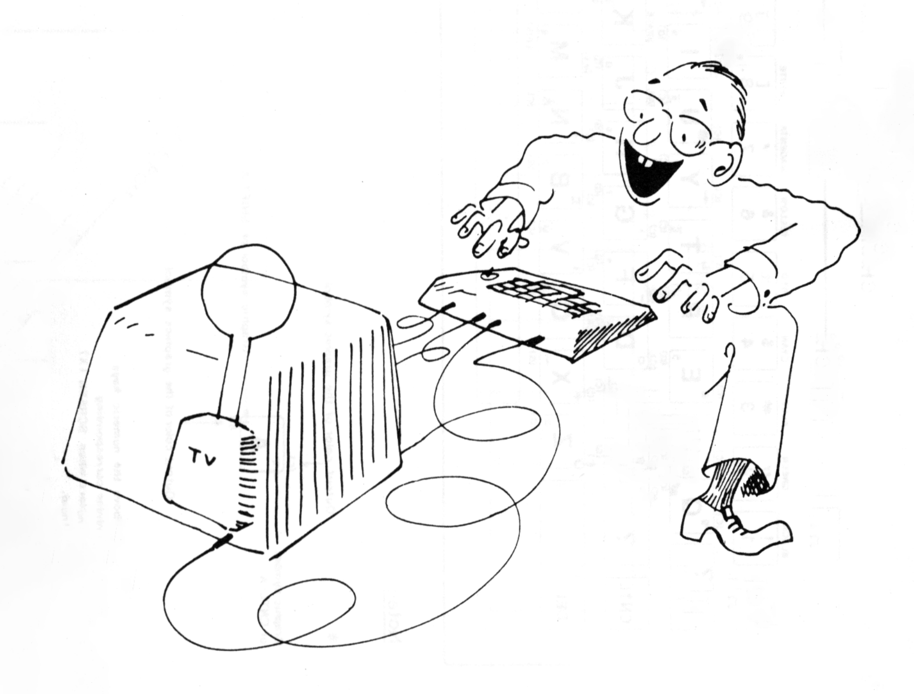
(II)SYNC

这是用于调整同步脉冲的振幅，一般不必调整这个控制，除非它被用戶调乱了，要再调整须用有经验的服务人员来做。

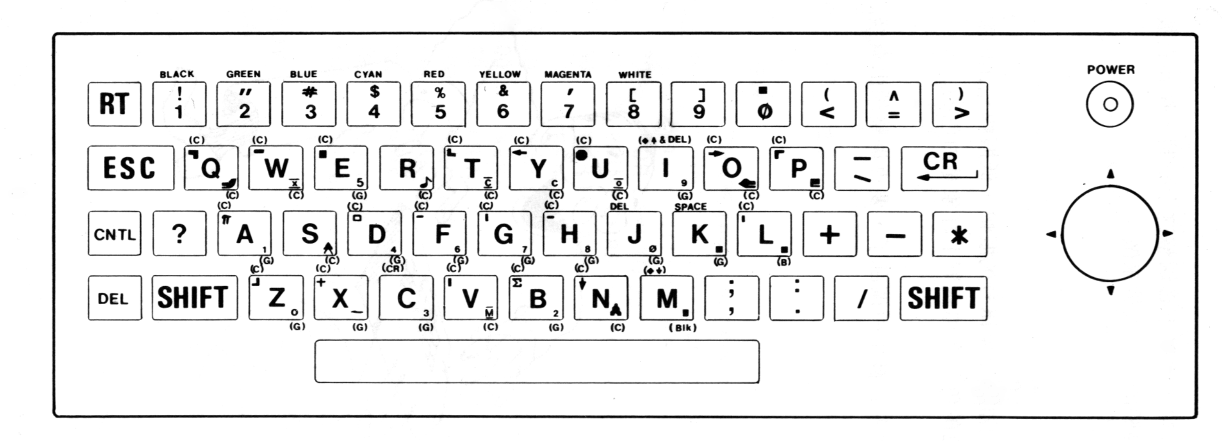
(III)VIDEO

这是用于调整送给ＲＦ调制器的见象讯号的振幅，低振幅给出的显示模糊不清，而高振幅会使黃色的字符在电视屏上好象是白色的。

一般不必调整这个控制，除非它被用戶调乱了，要再调整须由有经验的服务人员来做。



BUILT-IN CHARACTERS, A QUICK REFERENCE GUIDE



Note :

* Output color of the graphic symbol

Graphic symbol of CNTL-A

Graphic symbol of SHIFT-A

Output color of the graphics symbol

Key :	Letter	Color
	C	Cyan
	G	Green
	B	Blue
	Blk	Black

◆ : —— Cursor control

** Colors above the numeric keys indicate the corresponding screen colors when SCREEN (X) is executed.

A·6 存贮器图示

首	末		
	FFFF	显示屏 RAM	2K
F8ϕϕ	F7FF	字符 RAM	1K
F400	F3FF		1K
Fϕϕϕ	EFFF		4K
F4ϕϕ		保留	
Cϕϕϕ	DFFF		8K
BEϕϕ	BFFF	保留	
	BDFF	堆栈	
4ϕϕϕ		BASIC 程序 汇编程序 系统RAM 内部 32K RAM	
ϕϕϕϕ	3FFF	内部 16K ROM	扩充 ROM

存贮库 #

0 1 2 3 4 5 6 7

COMX－35 的存贮器图示

存贮单元	
0000－3FFF （十六进）	内部系统 ROM 包含(1)BASIC解释程序 (2)电视和键盘操作系统 (3)磁带操作系统
4000－43FF	系统参数存贮区。
4400－BDFF	BASIC程序和数据存贮区。 系统堆栈的顶部在BDFF。
BE00－BFFF	保留给磁盘操作系统。
C000－DFFF	8 K 字节的 8 个存贮库 －用于系统的扩展。 －在对某个区进引读或写操作之前程序必须先选择所要的存贮库。
E000－EFFF	保留 －用于存贮库之间的通信。 －不能适应于一个 8 K 字节存贮器的应用。
F000－F3FF	保留
F400－F7FF	字符定义的RAM区。 －这个区保持128个字符的点模式。 －字符可由BASIC的SHAPE指令改变。 －在PAL系统中每个字符由 9 个字节。 －在NTSC系统中每个字符用 8 个字节。 －这个区只能由转到CDP 1869 IC 的“字符存贮存取”的状态来访问。
F800－FFFF	显示屏页面存贮器 －每一字节保持屏上一个显示字符位的字符代码。 －在写入之前，F 800代表屏上左上角的字符位。 F 801是其下（在它右边）一个位置等等。 EBBF代表屏上右下角位置。 －硬件写入是通过改变在CDP 1869 IC里面的家庭地址寄存器完成。 －这个区是只写存贮区，而且只能在没有显示的时间访问，程序可以等到没有显示的时候才直接写入这个区。



